CREATE
COMPUTATIONAL RESEARCH AND ENGINEERING ACQUISITION TOOLS AND ENVIRONMENTS

Enhancing Non-Linear PDE Solver Convergence with Learnable Models

Advanced Modeling & Simulation (AMS) Seminar Series
NASA Ames Research Center, August 13th, 2026

Name: HPCMP CREATE
Controlled By: HPCMP CREATE
CUI Category:
Distribution: A
POC: Jennifer Abras, jennifer.n.abras.ctr@mail.mil

HPC MODERNIZATION PROGRAM

Dr. Jennifer Abras
HPCMP CREATE™-AV Helios Team, Bowhead Total Enterprise Solutions, LLC

Distribution Statement A. Approved for public release: distribution is unlimited.

Team Effort

- Abras J., Sitaraman J., and Hariharan N., "Enhancing Non-Linear PDE Solver Convergence with Learnable Models," 64th AIAA Aerospace Sciences Meeting, Orlando, FL, January 12-16, 2026
- Sitaraman J., Abras J., Hosseinverdi, S., and Hariharan N., "Enhancing Production-Oriented Non-Linear PDE Solver Convergence with Learnable Models" ICCFD 13, Milan, Italy, July 6-10, 2026
- Abras J., Sitaraman J., Hosseinverdi, S., "Enhancing Production PDE Solver Convergence with Learnable Models," 65th AIAA Aerospace Sciences Meeting, Orlando, FL, January 11-15, 2027

Team Effort

- Abras J., Sitaraman J., and Hariharan N., "Enhancing Non-Linear PDE Solver Convergence with Learnable Models," 64th AIAA Aerospace Sciences Meeting, Orlando, FL, January 12-16, 2026
- Sitaraman J., Abras J., Hosseinverdi, S., and Hariharan N., "Enhancing Production-Oriented Non-Linear PDE Solver Convergence with Learnable Models" ICCFD 13, Milan, Italy, July 6-10, 2026
- Abras J., Sitaraman J., Hosseinverdi, S., "Enhancing Production PDE Solver Convergence with Learnable Models," 65th AIAA Aerospace Sciences Meeting, Orlando, FL, January 11-15, 2027

Much has been completed, but this is still a work in progress

Literature Review

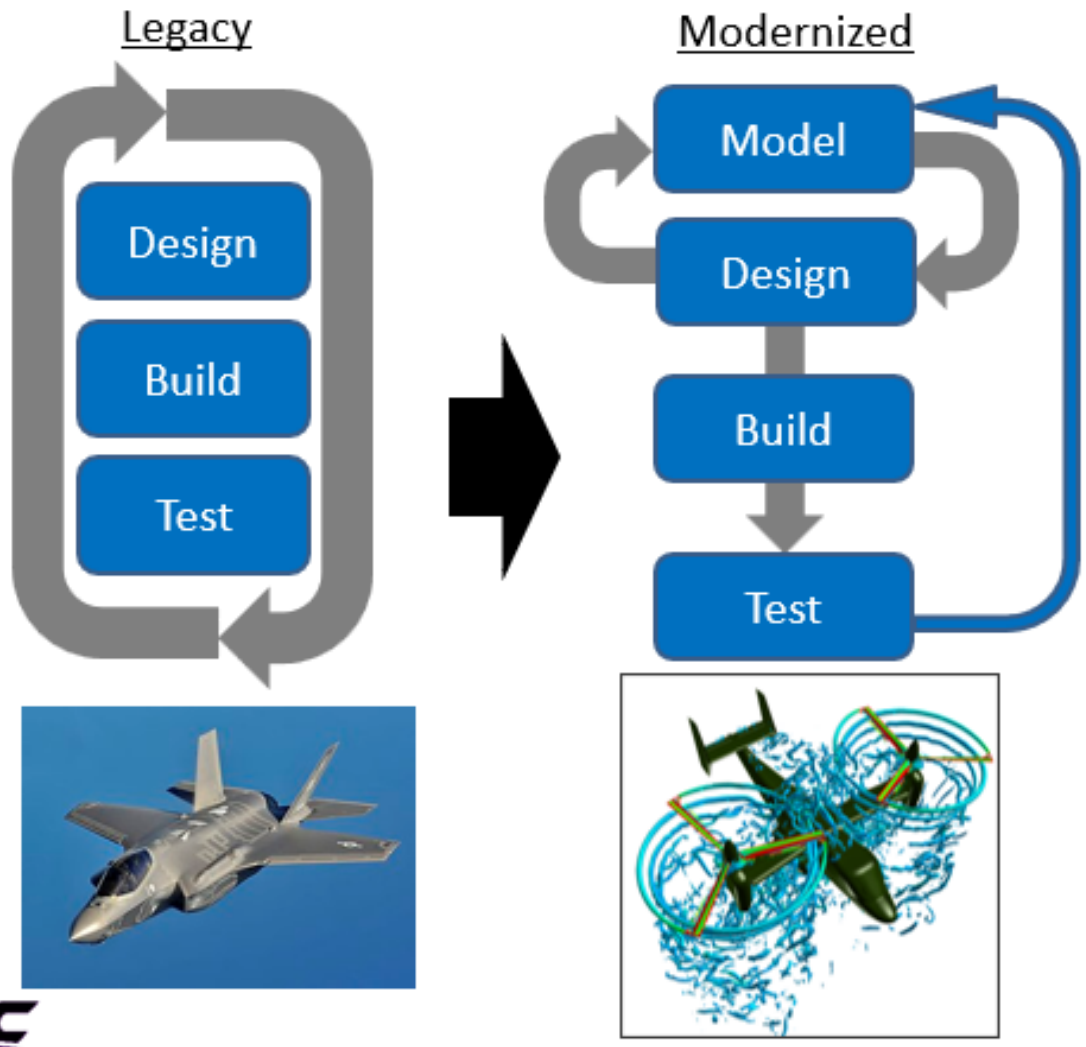
- Ajuria-Illarramendi, E., Alguacil, A., Bauerheim, M., Misdariis, A., Cuenot, B., and Benazera, E., "Towards a hybrid computational strategy based on Deep Learning for incompressible flows," AIAA 2020-3058. AIAA AVIATION 2020 FORUM, June 2020.
- Um, K., Brand, R., Fei, Y. R., Holl, P., and Thuerey, N., "Solver-in-the-loop: learning from differentiable physics to interact with iterative PDE-solvers," *Proceedings of the 34th International Conference on Neural Information Processing Systems*, Article 513, 2020, pp. 6111–6122.
- Vaupel, Y., Hamacher, N. C., Caspari, A., Mhamdi, A., Kevrekidis, I. G., and Mitsos, A., "Accelerating non-linear model predictive control through machine learning," *Journal of process control*, Vol. 92, August 2020, pp. 261–270.
- Luna, K., Klymko, K., and Blaschke, J., "Accelerating GMRES with Deep Learning in Real-Time," arXiv preprint arXiv:2103.10975 (2021).
- Huang, X., Ye, Z., Liu, H., Ji, S., Wang, Z., Yang, K., Li, Y., Wang, M., Chu, H., Yu, F., Hua, B., Chen, L., and Dong, B., "Meta-Auto-Decoder for Solving Parametric Partial Differential Equations," *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Red Hook, NY, Vol. 35, 2022, pp. 23426–23438.
- Nikolopoulos, S., Kalogeris, I., Papadopoulos, V., and Stavroulakis, G., "AI-enhanced iterative solvers for accelerating the solution of large-scale parametrized systems," arXiv preprint arXiv: 2207.02543 (2022).
- Arisaka, S., and Li, Q., "Principled Acceleration of Iterative Numerical Methods Using Machine Learning," *Proceedings of the 40th International Conference on Machine Learning*, Honolulu, Hawaii, Vol. 202, 2023, pp. 1041-1059.
- Saverio, L., "Accelerating Convergence of Linear Iterative Solvers using Machine Learning," Master Thesis, Tesi di Laurea Magistrale in Mathematical Engineering - Ingegneria Matematica, 2023.

Computational Research & Engineering Acquisition Tools and Environments (CREATE)

CREATE

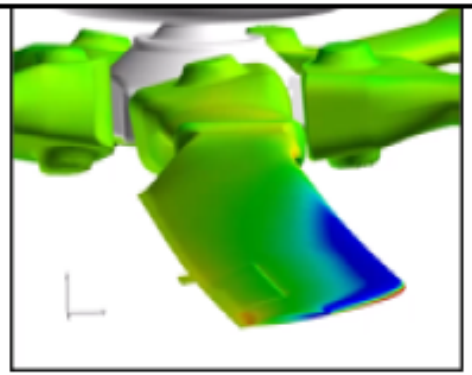
Software tools enabling the Department to exploit physics-based M&S for acquisition engineering, test and evaluation, and digital engineering/transformation

Enabling Digital Transformation

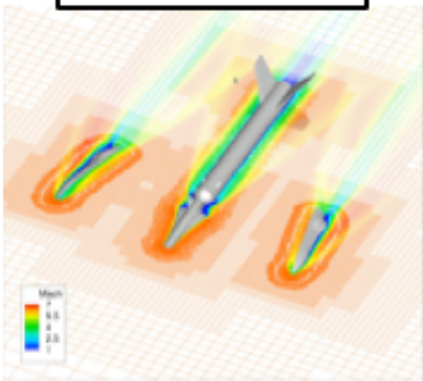


Multidisciplinary Emphasis

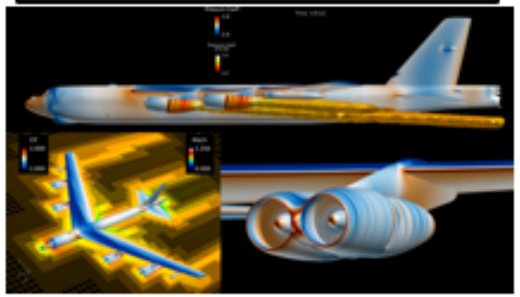
Aero-Structural Effects



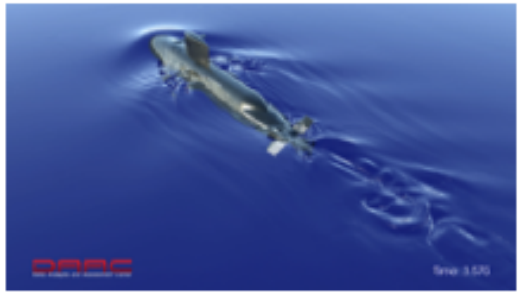
Hypersonics



Propulsion Integration



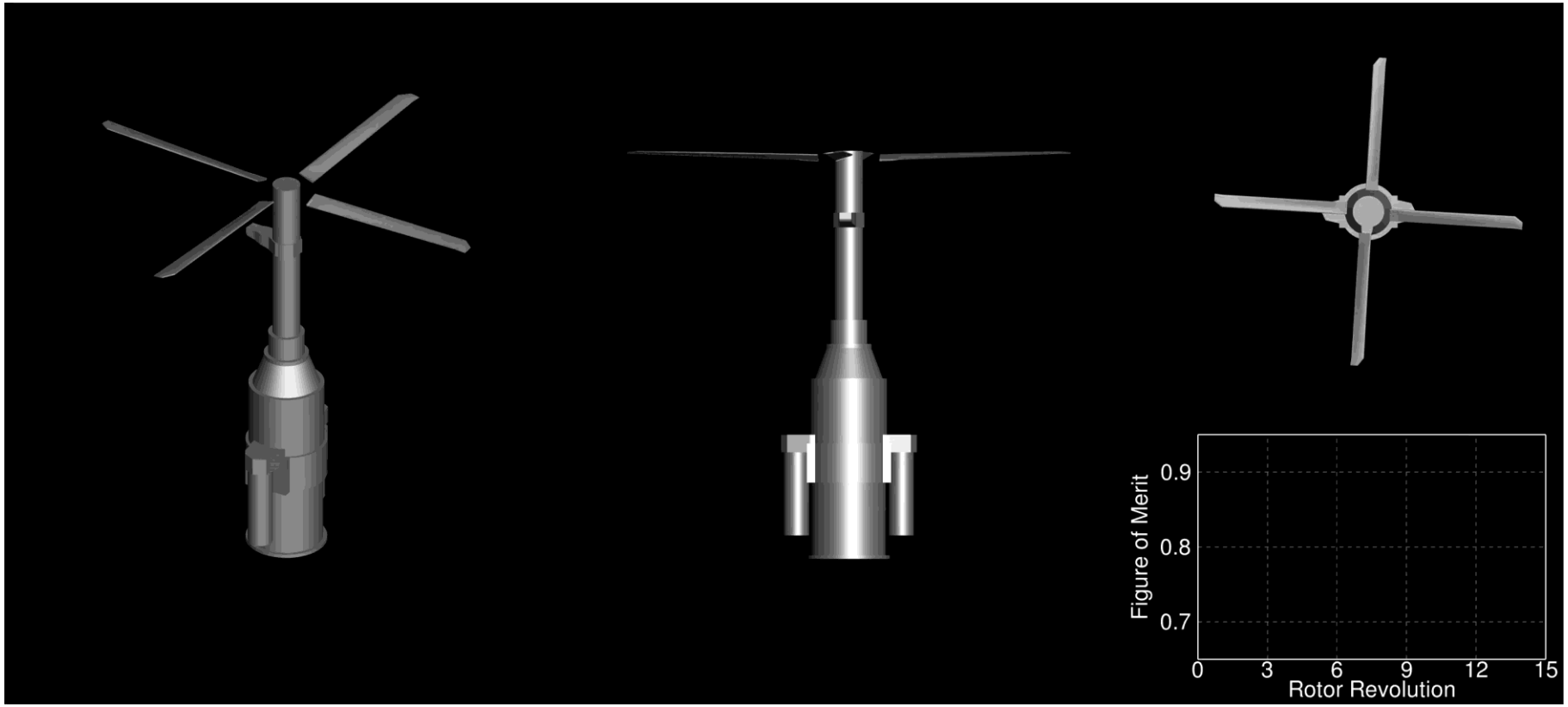
Multi-phase Flows



HPC

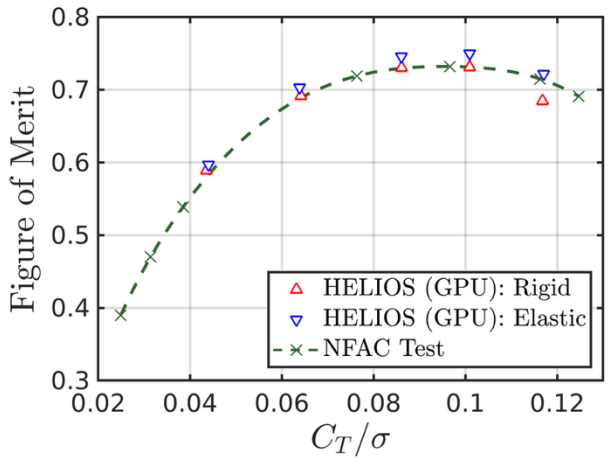
HPCMP CREATE™-AV Helios

- **Focuses on unsteady aeroelastic simulations of rotorcraft**

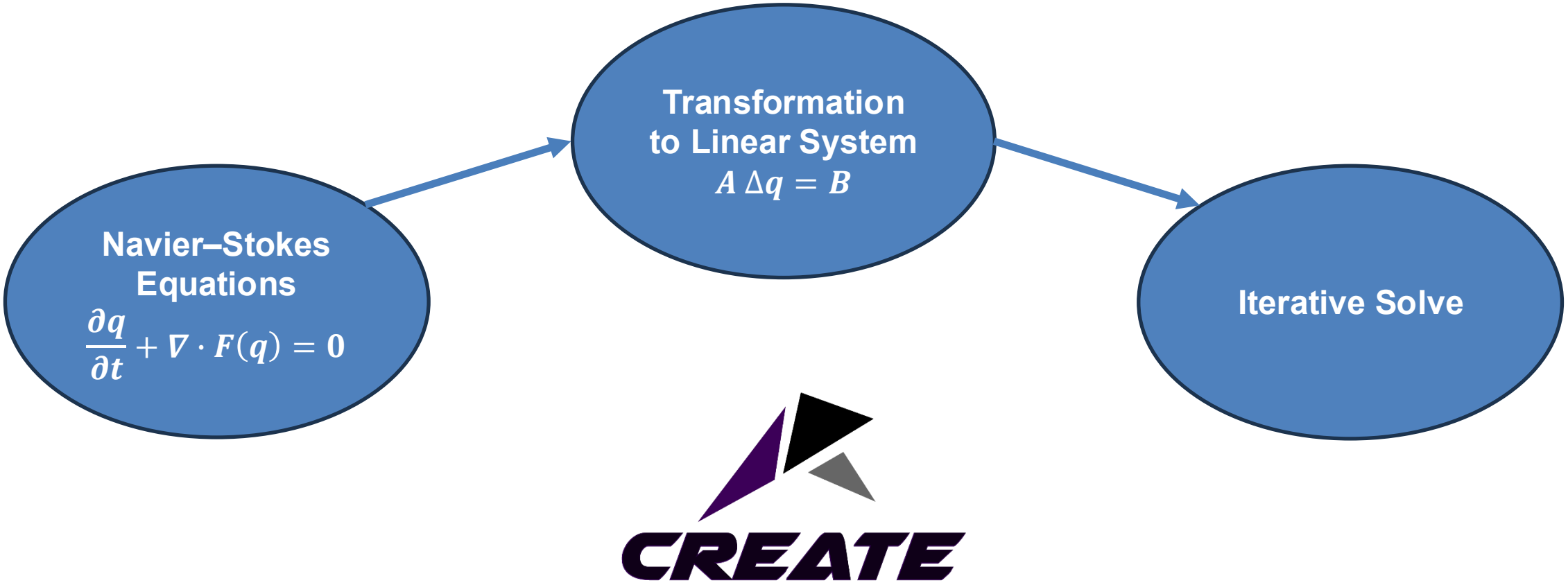


HVAB rotor simulation – experiment conducted in NASA Ames 80x120 tunnel

- Overset + adaptive meshing + fluid/structure coupling
- Performance portable (GPU and CPU)
- Current performance: 140M non-linear updates/second on 1 H100

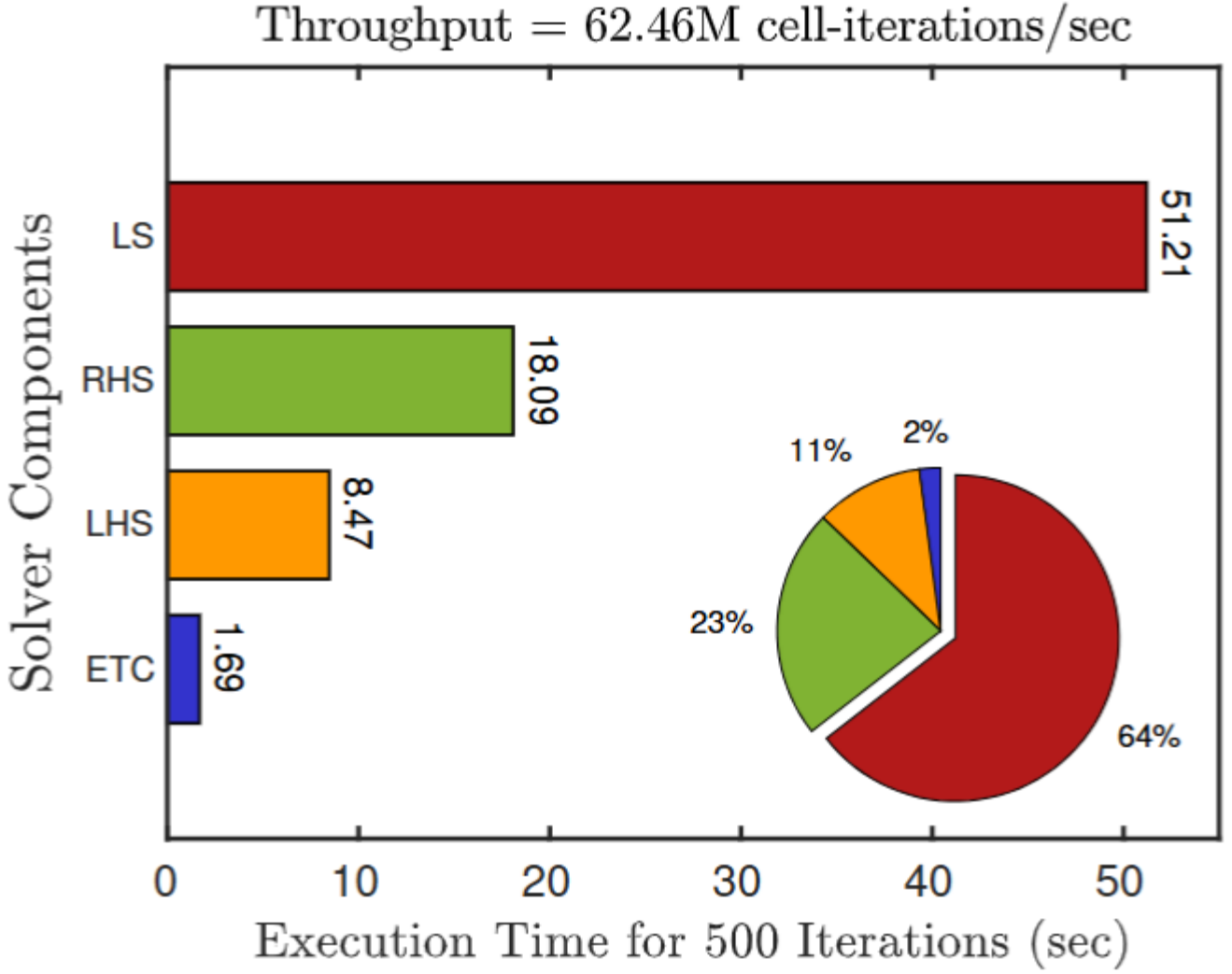


Production Non-Linear PDE Solvers



Solver Timing Breakdown

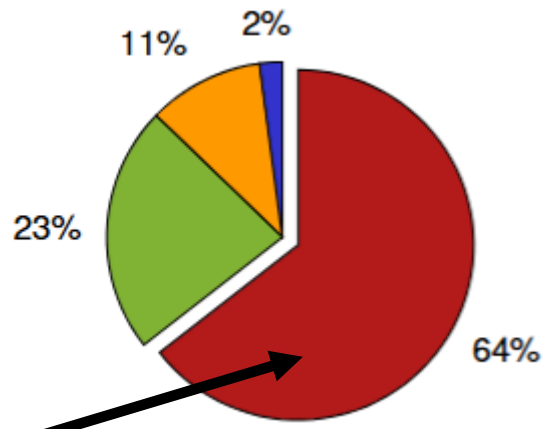
- Unsteady, time-dependent, implicit simulations: most time is spent in the linear solver
- Implementing methods to reduce the number of linear solver calls potentially saves significant time



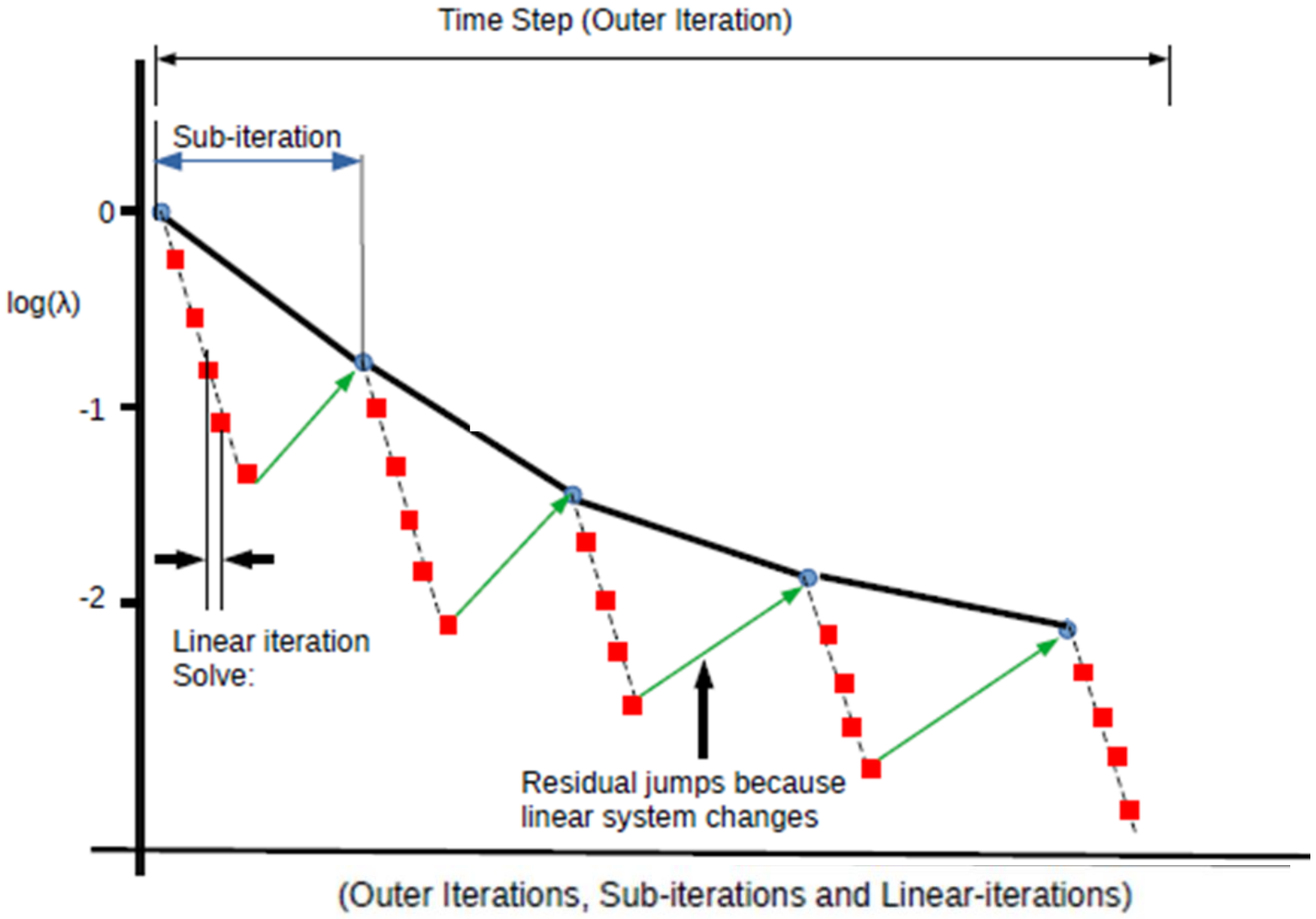
Shirzad Hosseinverdi, Jay Sitaraman, Dylan Jude, Pavithra Premaratne, Lucas Erlandson and David Appelhans. "Rapidus: Performance-Portable Parallel Flow Solver for Aerospace Applications," AIAA 2025-1484. AIAA SCITECH 2025 Forum. January 2025.

Time-Marching Algorithm

Time Stepping Loop:
- **Non-Linear Subiteration Loop:**
  - Compute Temporal Contribution
  - Compute Residual
  - Check Non-Linear Convergence (**Break**)
  - Compute Jacobian
  - Form Linear System
  - Spilt A into Sparse Components
  - Initialize Linear Increment
  - **Linear Solver Loop:**
    - Compute Linear Increment
    - Check Linear Convergence (**Break**)
    - Update Increment
  - Update Solution
- Update Previous Time Steps

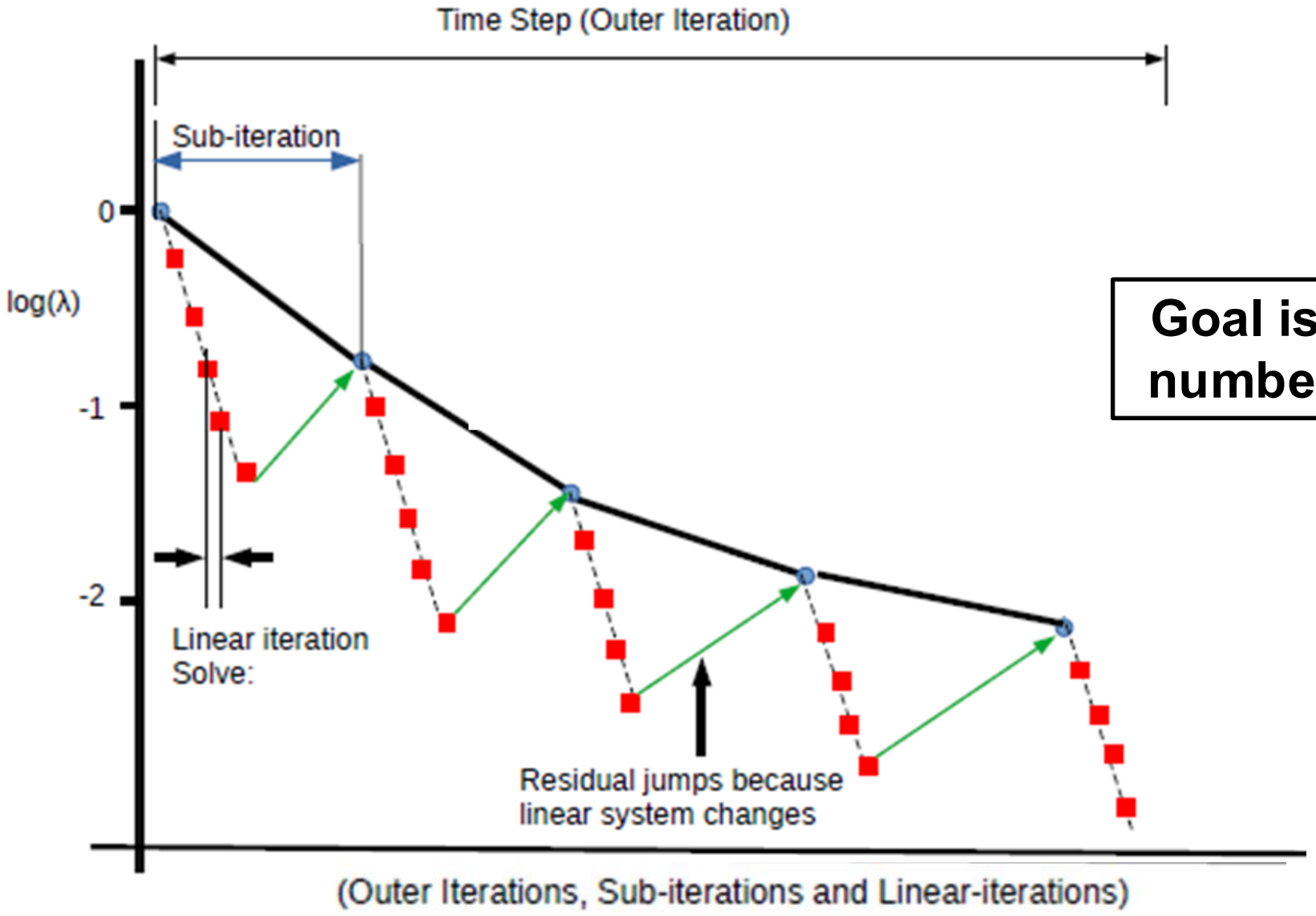


Time-Marching Algorithm Residual Illustration



Abras J., Sitaraman J., and Hariharan N., "Enhancing Non-Linear PDE Solver Convergence with Learnable Models," 64th AIAA Aerospace Sciences Meeting, Orlando, FL, January 12-16, 2026

Time-Marching Algorithm Residual Illustration



Goal is to reduce the number of red points

Abras J., Sitaraman J., and Hariharan N., "Enhancing Non-Linear PDE Solver Convergence with Learnable Models," 64th AIAA Aerospace Sciences Meeting, Orlando, FL, January 12-16, 2026

ML Augmented Algorithm

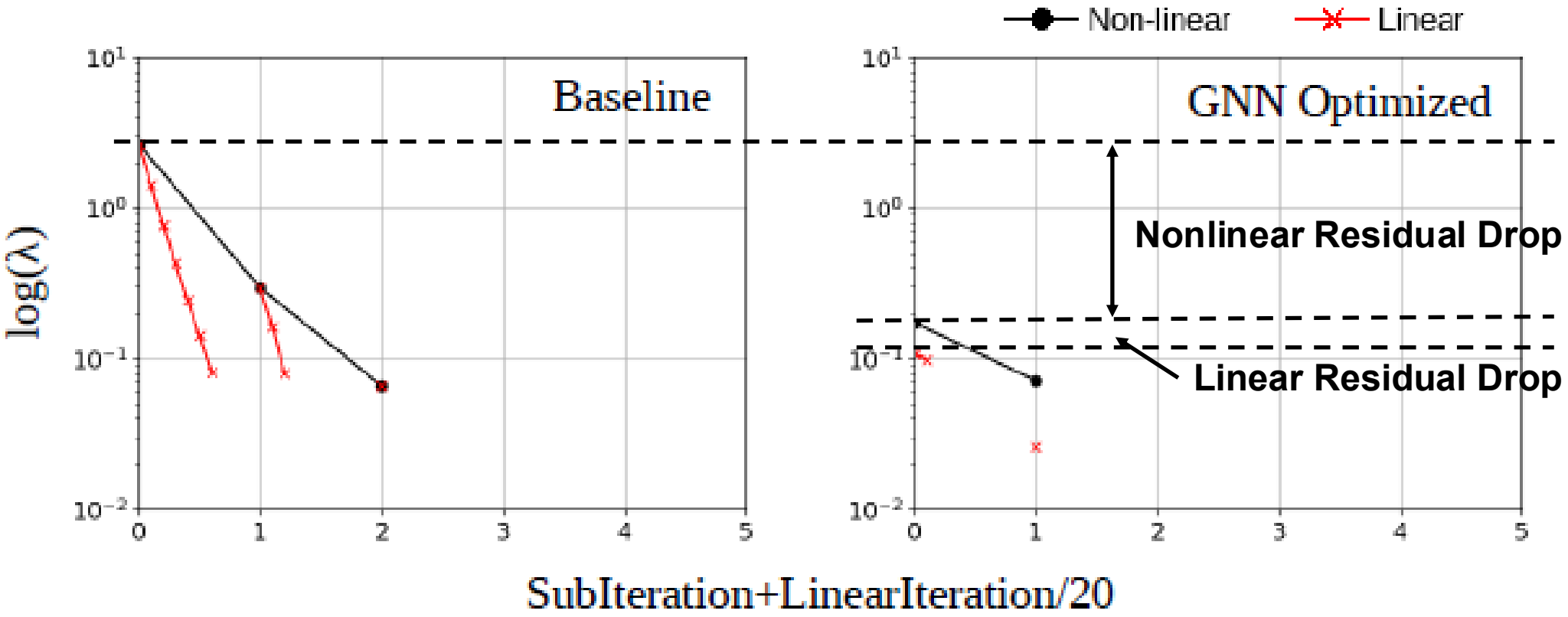
Algorithm 2 Reflex: ML-enhanced Time-Marching

1: Initialize Q^0, Q^n, Q^{n-1}; initialize smodel, lmodel
2: **for** $n = 1$ to N_{steps} **do**
3: Compute $R \leftarrow S(Q) + \text{TemporalSource}(Q, Q^n, Q^{n-1})$
4: **if** smodel.can_predict() **then**
5: $\Delta Q \leftarrow \text{smodel}(R)$; $Q \leftarrow Q + \Delta Q$; recompute R
6: **for** $s = 1$ to N_{subiter} **do**
7: Recompute R; **if** $\|R\|/\|R_0\| < \epsilon_{\text{nl}}$ **break**
8: Assemble $D \leftarrow \text{DiagJacobian}(Q)$; $B \leftarrow -R$
9: **if** lmodel.can_predict() **then**
10: $X \leftarrow \text{lmodel}(B)$
11: **Residual guard:** while $\|B - AX\| > \|B\|$ and trials < 7: $X \leftarrow X/2$
12: **if** guard failed: $X \leftarrow 0$
13: **else**
14: $X \leftarrow 0$
15: Solve $(D^{-1}A)\,\delta X = D^{-1}(B - AX)$ via GMRES(m) or Block-Jacobi(N_{sw}); $X \leftarrow X + \delta X$
16: $Q \leftarrow Q + X$
17: Curate (B, X) for lmodel training
18: $Q^{n-1} \leftarrow Q^n$; $Q^n \leftarrow Q$
19: Curate $(R_0, \Delta Q_{\text{total}})$ for smodel training
20: Retrain smodel and lmodel if error threshold exceeded

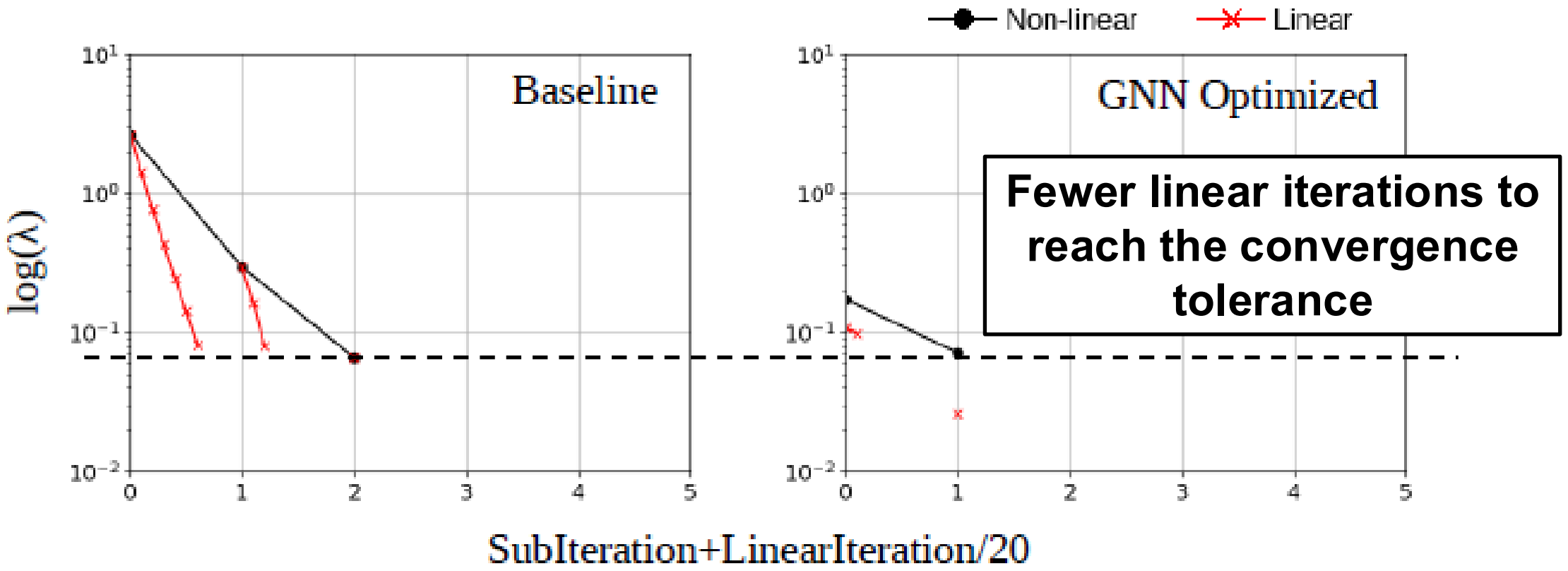
Warm Start Non-Linear Iterations

Warm Start Linear Iterations

Warm Start Residual Impact



Convergence Tolerance



ML Training Features

Algorithm 2 Reflex: ML-enhanced Time-Marching

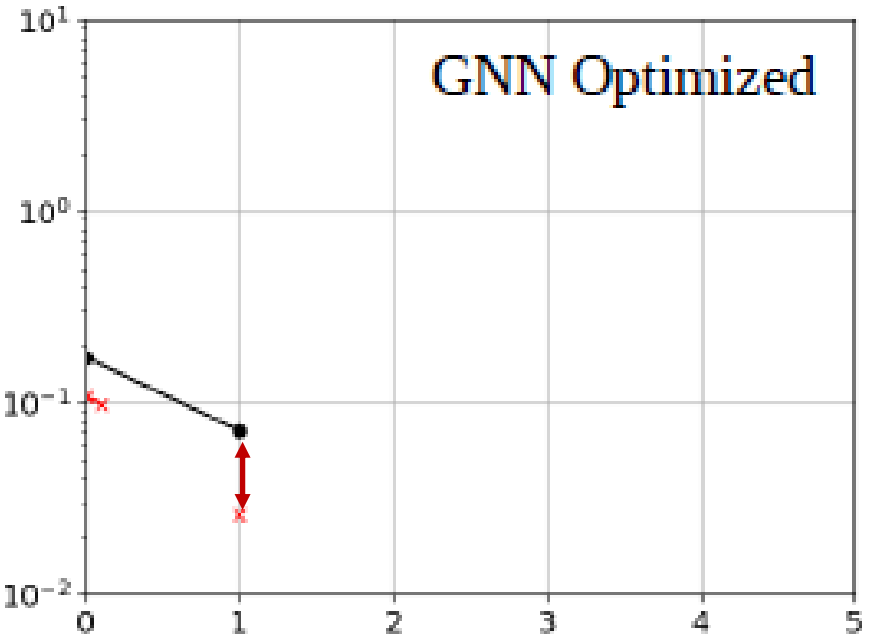
1: Initialize Q^0, Q^n, Q^{n-1}; initialize smodel, lmodel
2: **for** $n = 1$ to N_{steps} **do**
3: Compute $R \leftarrow S(Q) + \text{TemporalSource}(Q, Q^n, Q^{n-1})$
4: **if** smodel.can_predict() **then**
5: $\Delta Q \leftarrow \text{smodel}(R)$; $Q \leftarrow Q + \Delta Q$; recompute R
6: **for** $s = 1$ to N_{subiter} **do**
7: Recompute R; **if** $\|R\|/\|R_0\| < \epsilon_{\text{nl}}$ **break**
8: Assemble $D \leftarrow \text{DiagJacobian}(Q)$; $B \leftarrow -R$
9: **if** lmodel.can_predict() **then**
10: $X \leftarrow \text{lmodel}(B)$
11: **Residual guard:** while $\|B - AX\| > \|B\|$ and trials < 7: $X \leftarrow X/2$
12: **if** guard failed: $X \leftarrow 0$
13: **else**
14: $X \leftarrow 0$
15: Solve $(D^{-1}A)\,\delta X = D^{-1}(B - AX)$ via GMRES(m) or Block-Jacobi(N_{sw}); $X \leftarrow X + \delta X$
16: $Q \leftarrow Q + X$
17: Curate (B, X) for lmodel training
18: $Q^{n-1} \leftarrow Q^n$; $Q^n \leftarrow Q$
19: Curate $(R_0, \Delta Q_{\text{total}})$ for smodel training
20: Retrain smodel and lmodel if error threshold exceeded

Data Curation and Training

ML Residual Guard

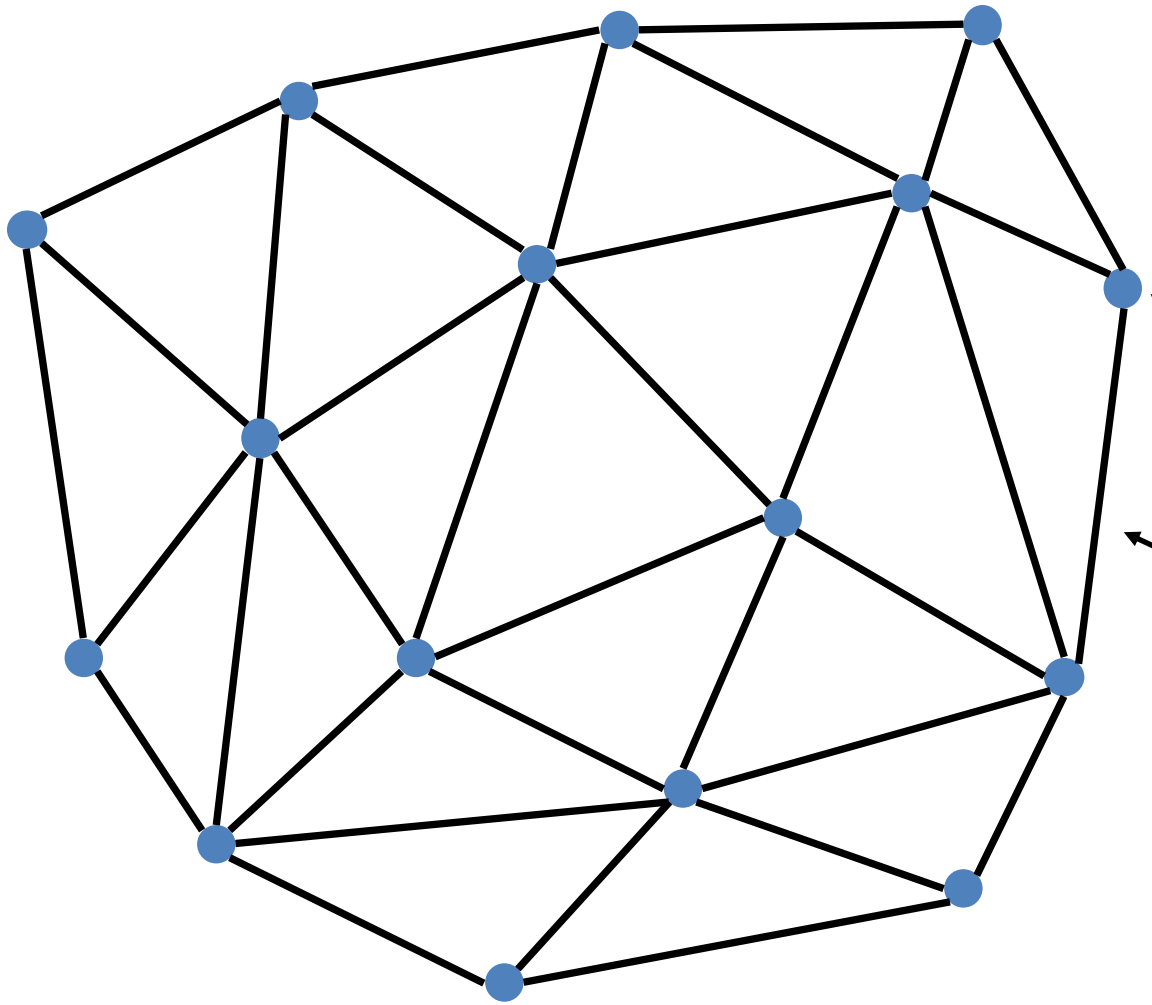
Algorithm 2 Reflex: ML-enhanced Time-Marching

Initialize Q^0, Q^n, Q^{n-1}; initialize smodel, lmodel
for $n = 1$ to N_{steps} **do**
  Compute $R \leftarrow S(Q) + \text{TemporalSource}(Q, Q^n, Q^{n-1})$
  if smodel.can_predict() **then**
    $\Delta Q \leftarrow \text{smodel}(R)$; $Q \leftarrow Q + \Delta Q$; recompute R
  for $s = 1$ to N_{subiter} **do**
    Recompute R; **if** $\|R\|/\|R_0\| < \epsilon_{\text{nl}}$ **break**
    Assemble $D \leftarrow \text{DiagJacobian}(Q)$; $B \leftarrow -R$
    if lmodel.can_predict() **then**
      $X \leftarrow \text{lmodel}(B)$
      Residual guard: while $\|B - AX\| > \|B\|$ and trials < 7: $X \leftarrow X/2$
      if guard failed: $X \leftarrow 0$
    else
      $X \leftarrow 0$
    Solve $(D^{-1}A)\,\delta X = D^{-1}(B - AX)$ via GMRES(m) or Block-Jacobi(N_{sw}); $X \leftarrow X + \delta X$
    $Q \leftarrow Q + X$
    Curate (B, X) for lmodel training
  $Q^{n-1} \leftarrow Q^n$; $Q^n \leftarrow Q$
  Curate $(R_0, \Delta Q_{\text{total}})$ for smodel training
  Retrain smodel and lmodel if error threshold exceeded



Only Apply Linear Warm Start if it Helps

Graph Neural Network: Node-Centered Solver

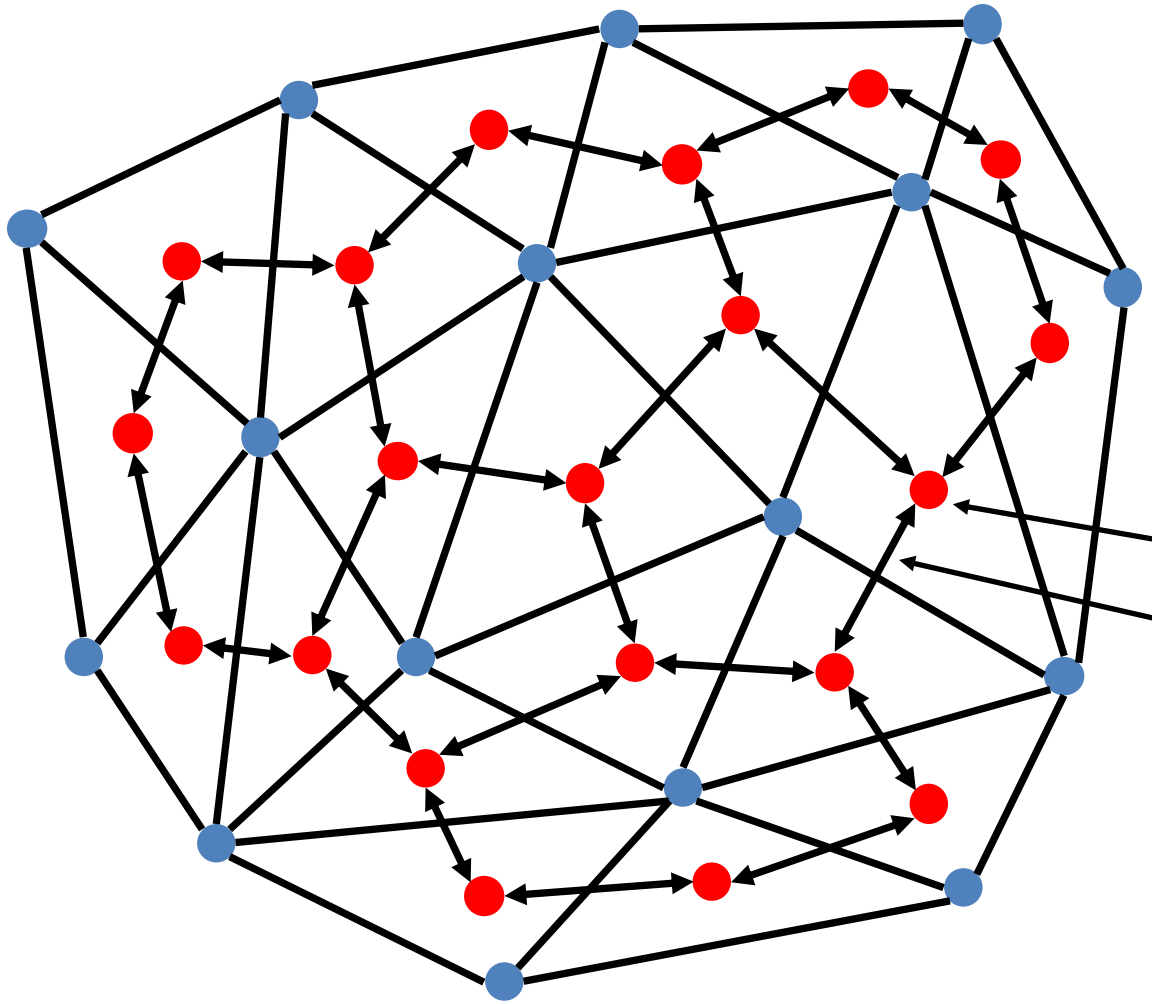


Graph Neural Networks are a Natural Fit for Unstructured CFD methods

CFD Grid Nodes = Graph Nodes

CFD Grid Edges = Graph Edges

Graph Neural Network: Cell-Centered Solver



For Cell-Centered CFD Methods the Graph Nodes are the Cell-Centers and the Graph Edges are Derived from the Face Connectivity

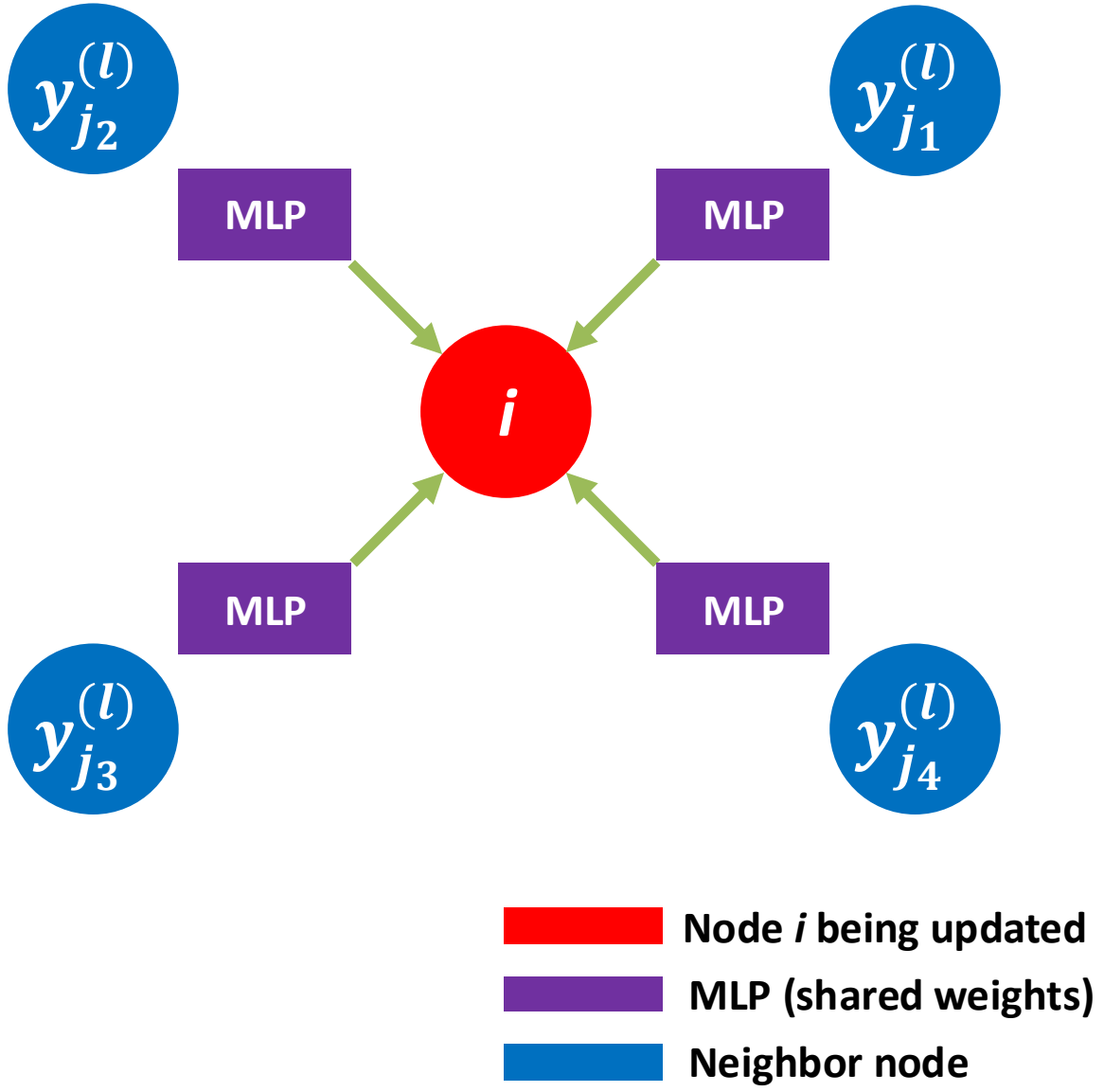
Graph Nodes

Graph Edges

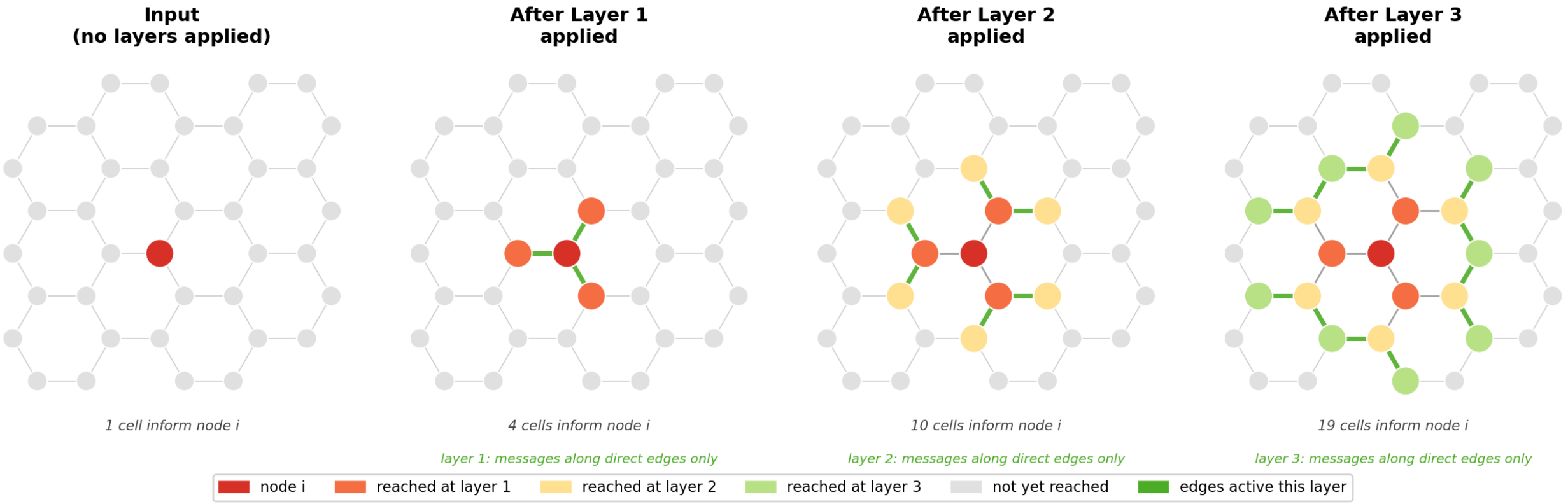
Graph Neural Network Layer

$$y_i^{(l+1)} = y_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \mathrm{MLP}(y_j^{(l)})$$

- Every layer has the same structure
- Nodes communicate to their neighbors using edge weights and biases and passed through an activation function
- Mirrors unstructured mesh graph iterative updates



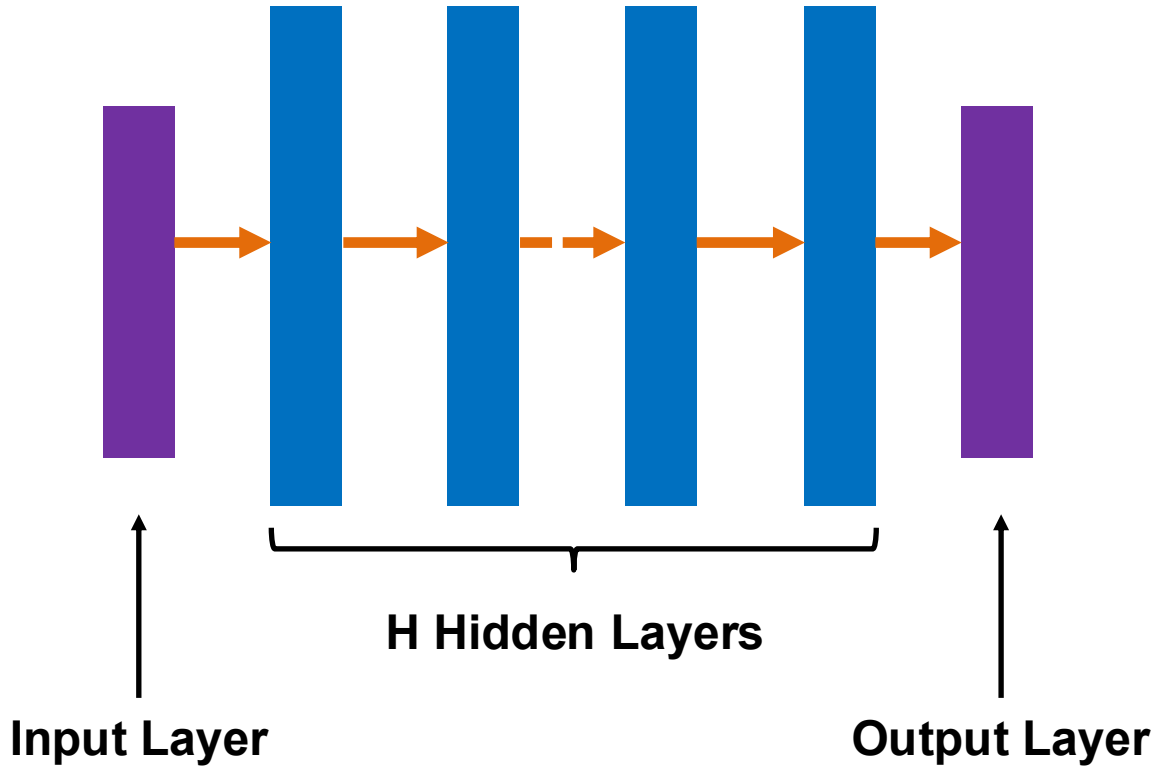
Graph Neural Network Message Passing



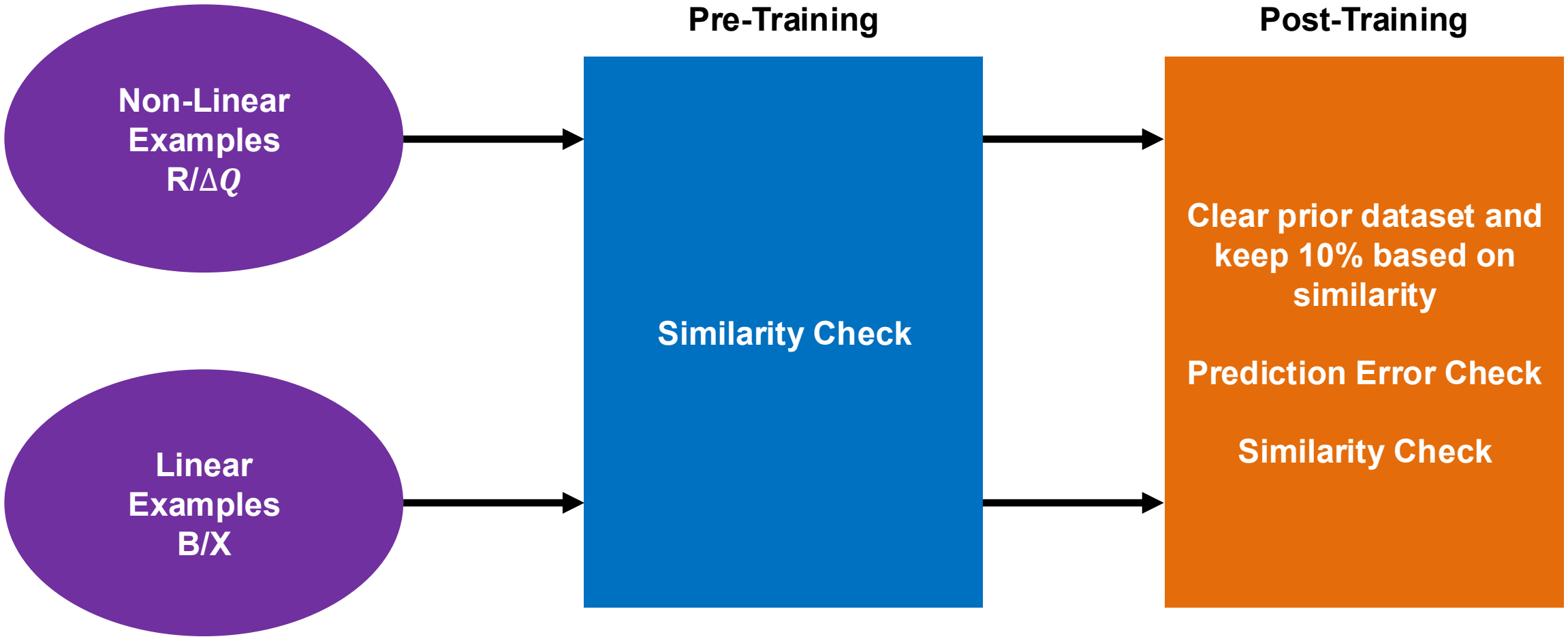
Number of hidden layers controls the radius of influence

Graph Neural Network Architecture

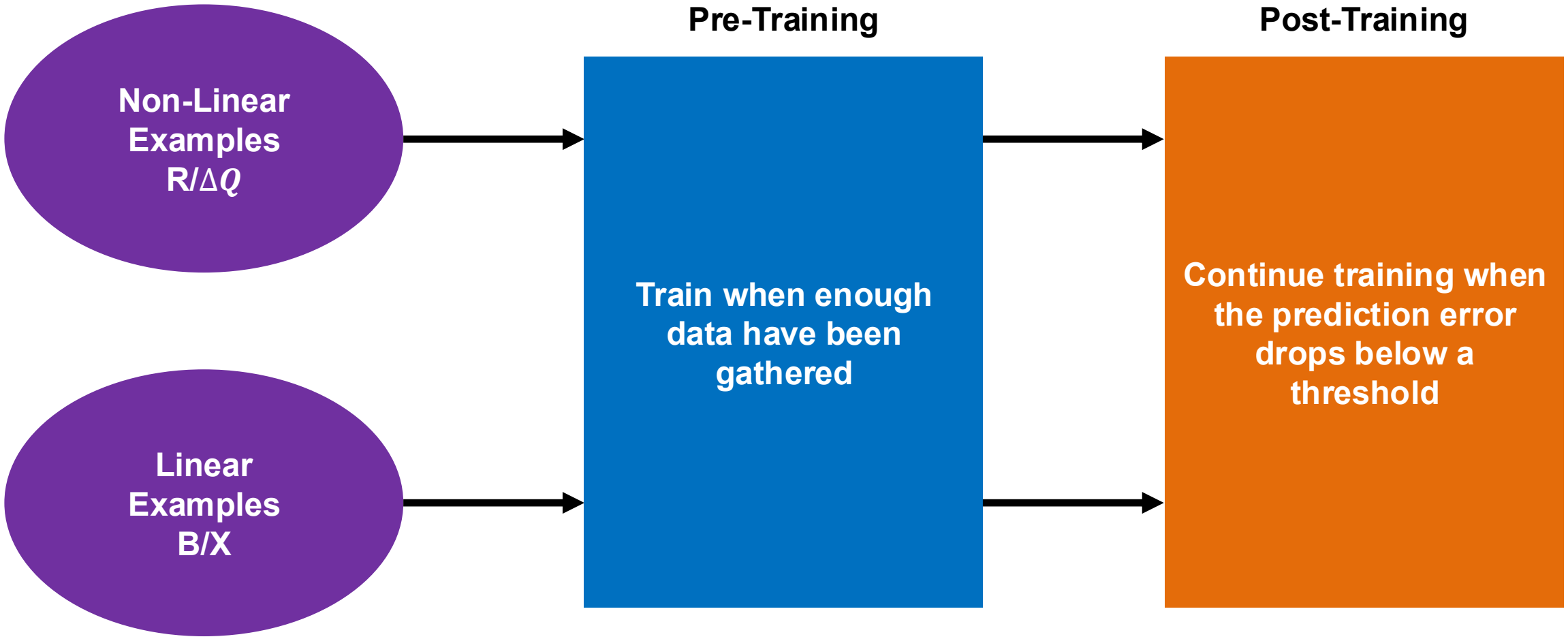
$$\text{GraphSolver}(B) = \text{Proj}_{\text{out}} \circ [\sigma \circ \text{GNNLayer}]^{H} \circ \text{Proj}_{\text{in}}(B)$$



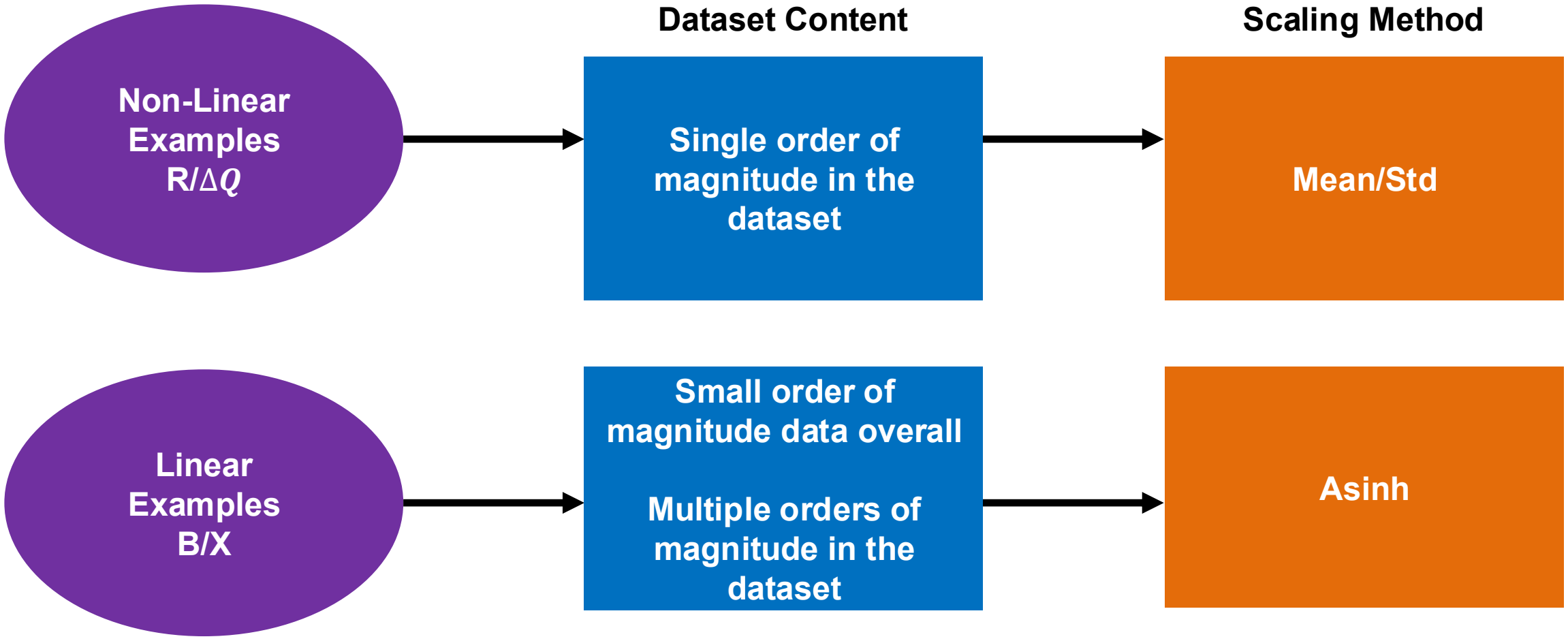
GNN Solver On-Board Training: Gathering Data



GNN Solver On-Board Training: Training



GNN Solver On-Board Training: Data Scaling



Production Level Long Term Goal

Time saved in the solver > Time used to train and execute the GNN

Keep in mind that the predictions only need to be good enough

1. **Keep the dataset small to reduce training time**
2. **Make sure the dataset content is effective**
3. **Train to a good enough loss level**
4. **Retrain only when needed**

Canonical PDEs

The benefits of the GNN integrated solver are demonstrated on 3 different equation sets of increasing complexity

Equation	Field ($\mathbf{q}$)	Flux $\mathbf{F}(\mathbf{q})$
Inviscid Burgers	u	$\frac{1}{2}u^2$
1-D Euler Equation	$\{\rho, \rho u, e\}$	$\{\rho u, \rho u^2 + p, (e+p)u\}$
2-D Euler Equation	$\{\rho, \rho\vec{u}, e\}$	$\begin{Bmatrix} \rho\vec{u} \\ \rho\vec{u} \otimes \vec{u} + p\mathbf{I} \\ (e+p)\vec{u} \end{Bmatrix}$

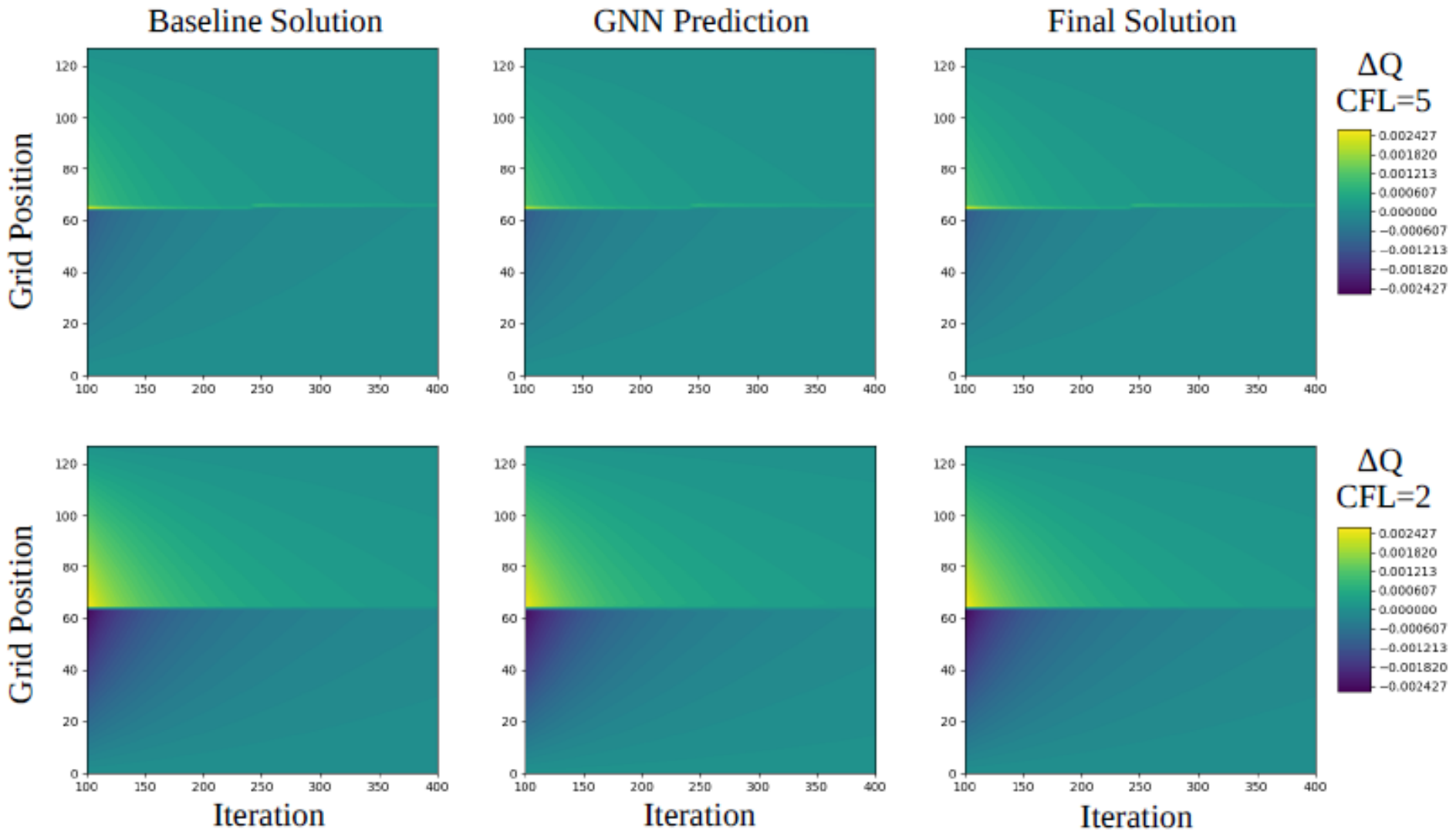
Abras J., Sitaraman J., and Hariharan N., "Enhancing Non-Linear PDE Solver Convergence with Learnable Models," 64th AIAA Aerospace Sciences Meeting, Orlando, FL, January 12-16, 2026

1D Burgers: Problem Setup

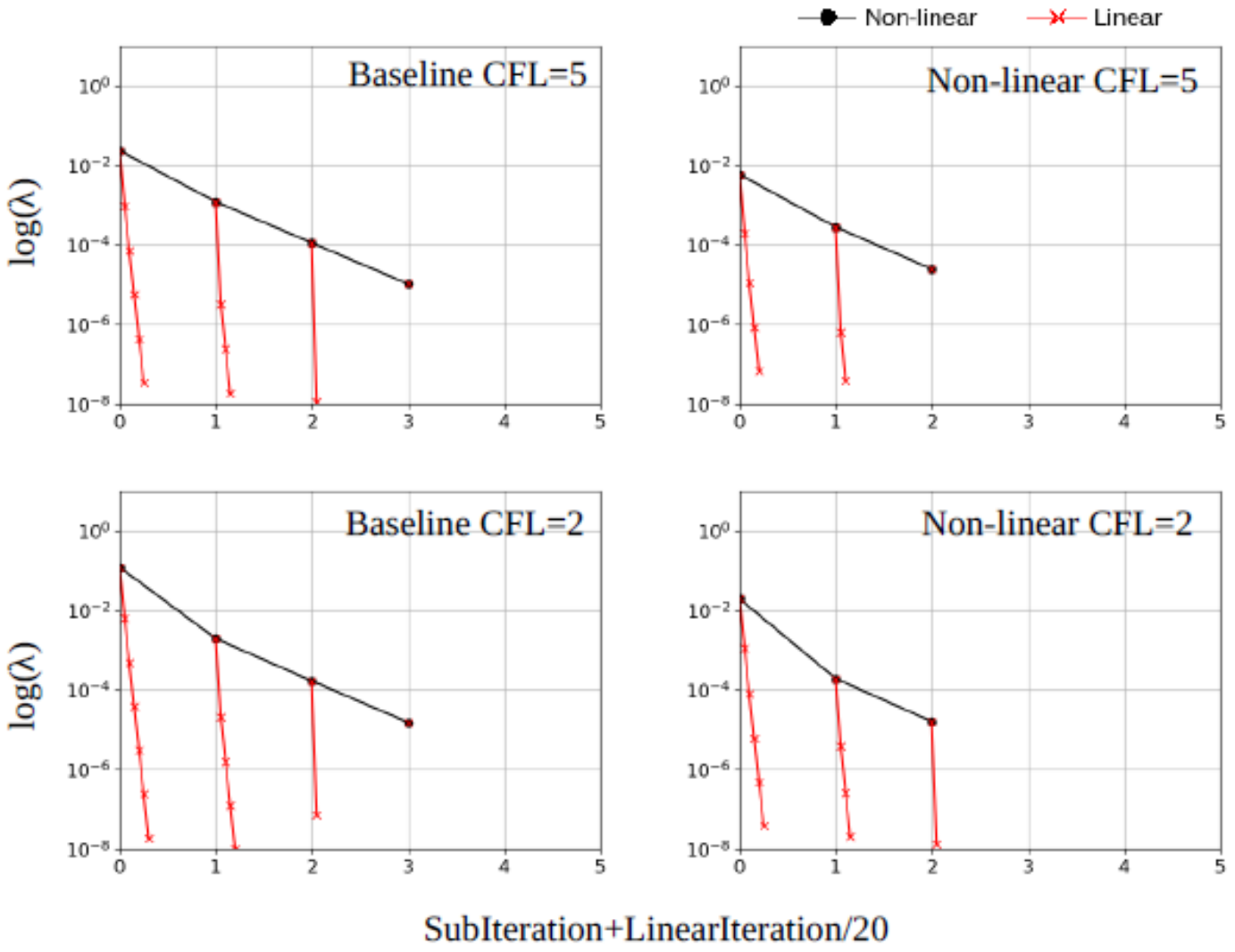
- 1D grid of 128 nodes
- Periodic boundary conditions
- Initialized with a sinusoid
- Compare CFL=5 and CFL=2

Initially we just look at the nonlinear solver warm start

1D Burgers: Contour Plots



1D Burgers: Residuals, Iteration 400



30.8% savings

20.0% savings

1D Burgers: Conclusions

GNN predictions are reasonable

Modest reduction in nonlinear residuals

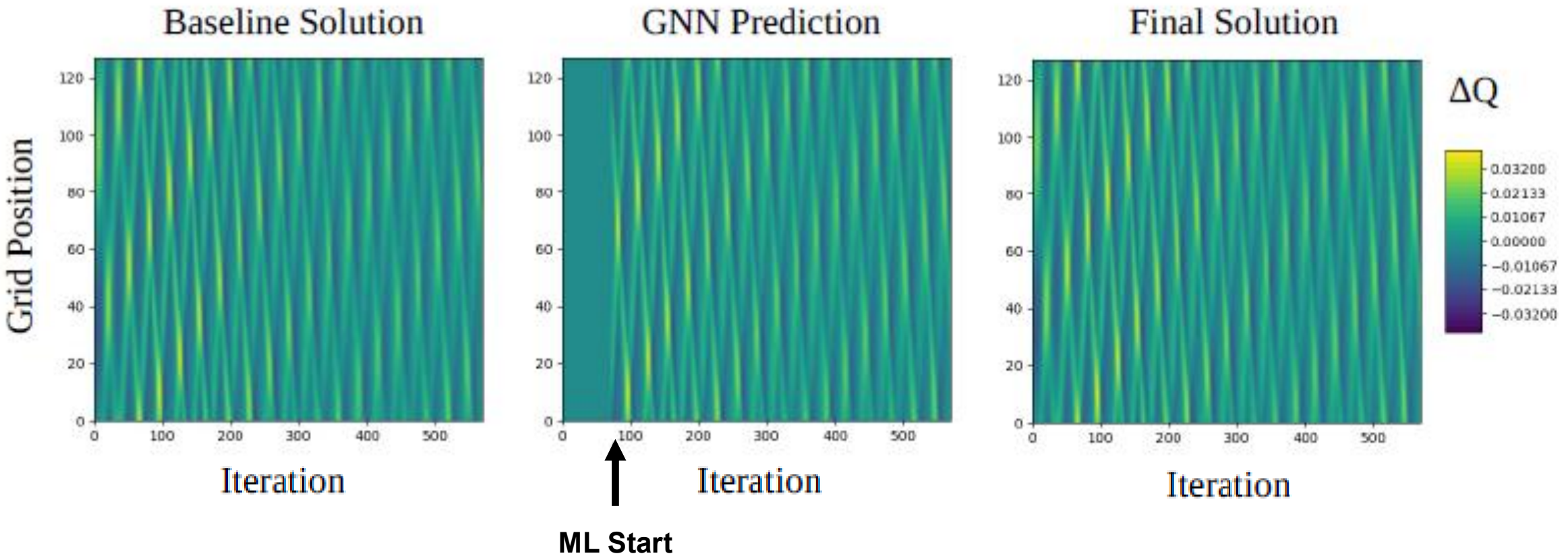
The initial implementation gains are modest, but show that we are heading in the right direction

1D Euler: Problem Setup

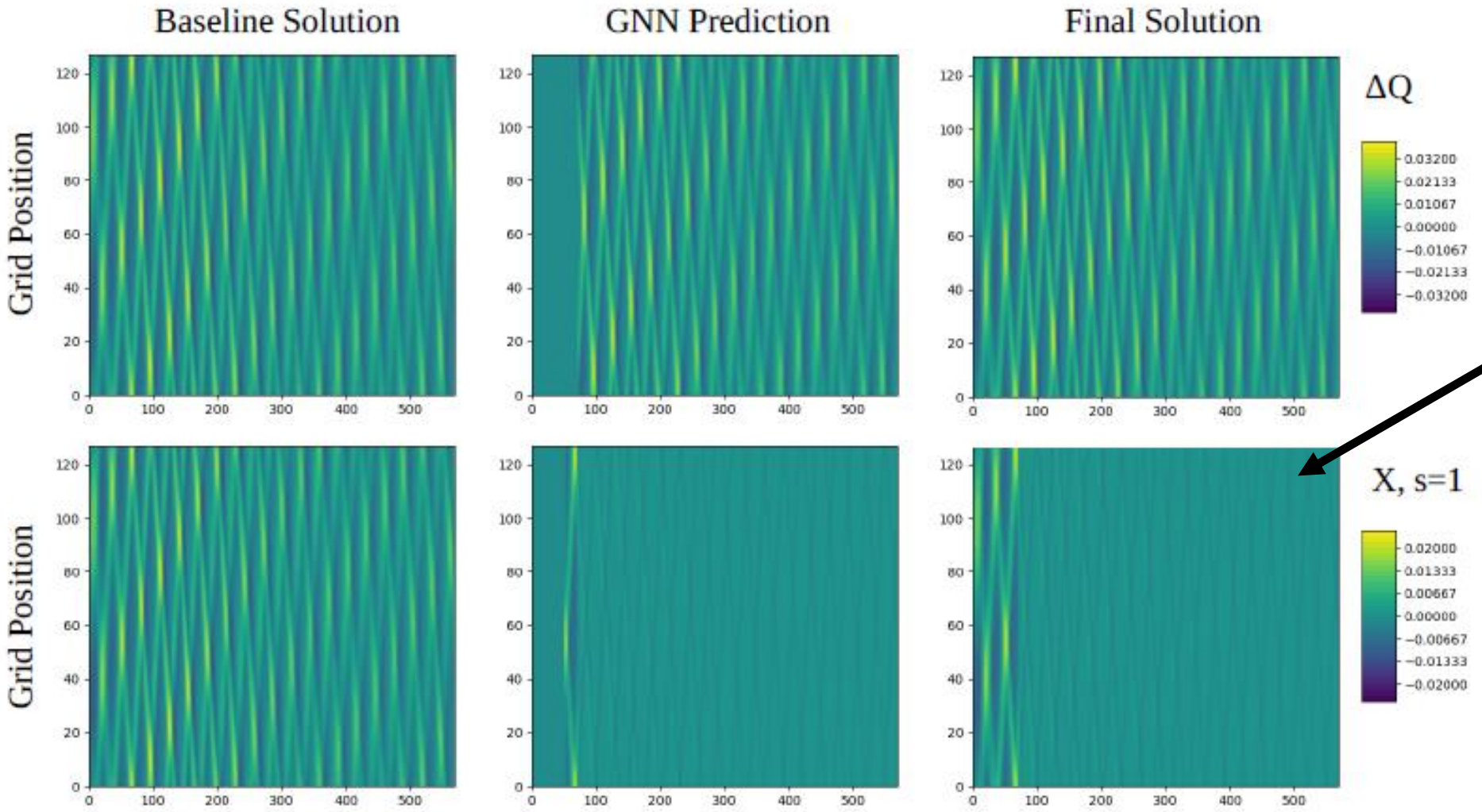
- 1D grid of 128 nodes
- Periodic boundary conditions
- CFL = 5
- Initialized using
  - $\rho = 1 + 0.1sin(2\pi x)$
  - $p = \rho^{\gamma}/\gamma$
  - $u = 0.1$

Increasing the complexity to 3 equations and adding the linear solver warm start

1D Euler: Contour Plots, Non-Linear Only

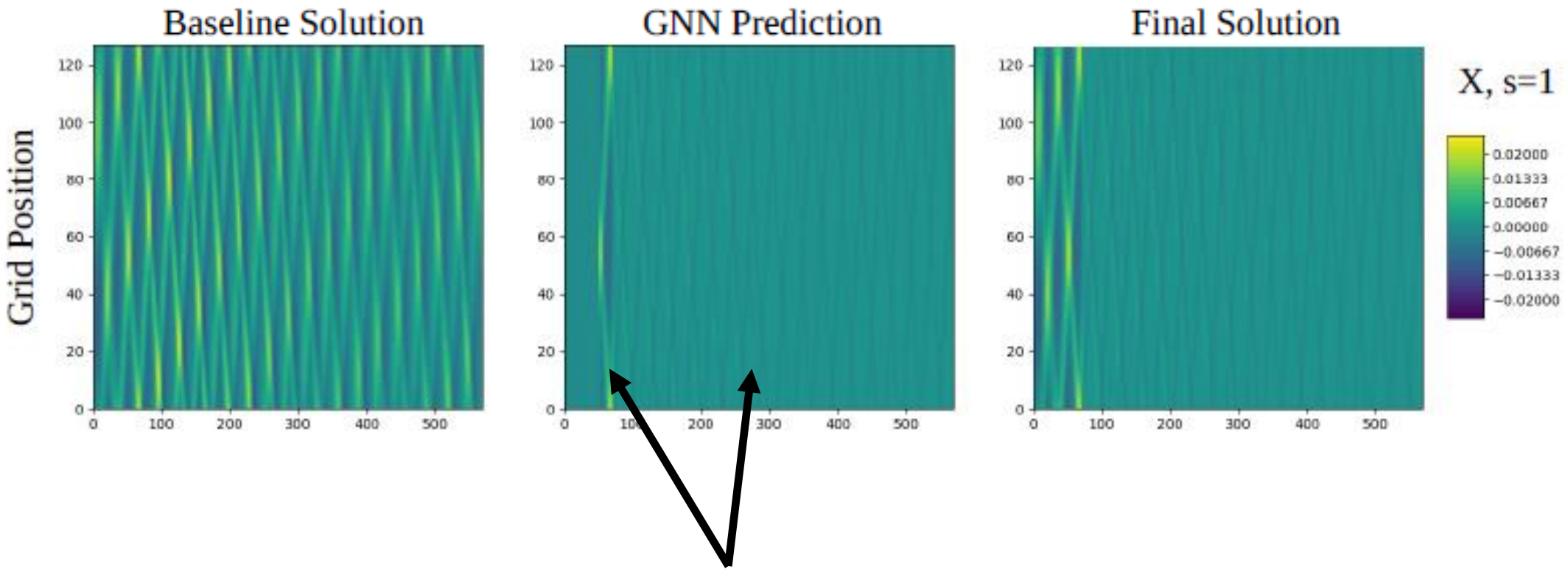


1D Euler: Contour Plots, Non-Linear + Linear



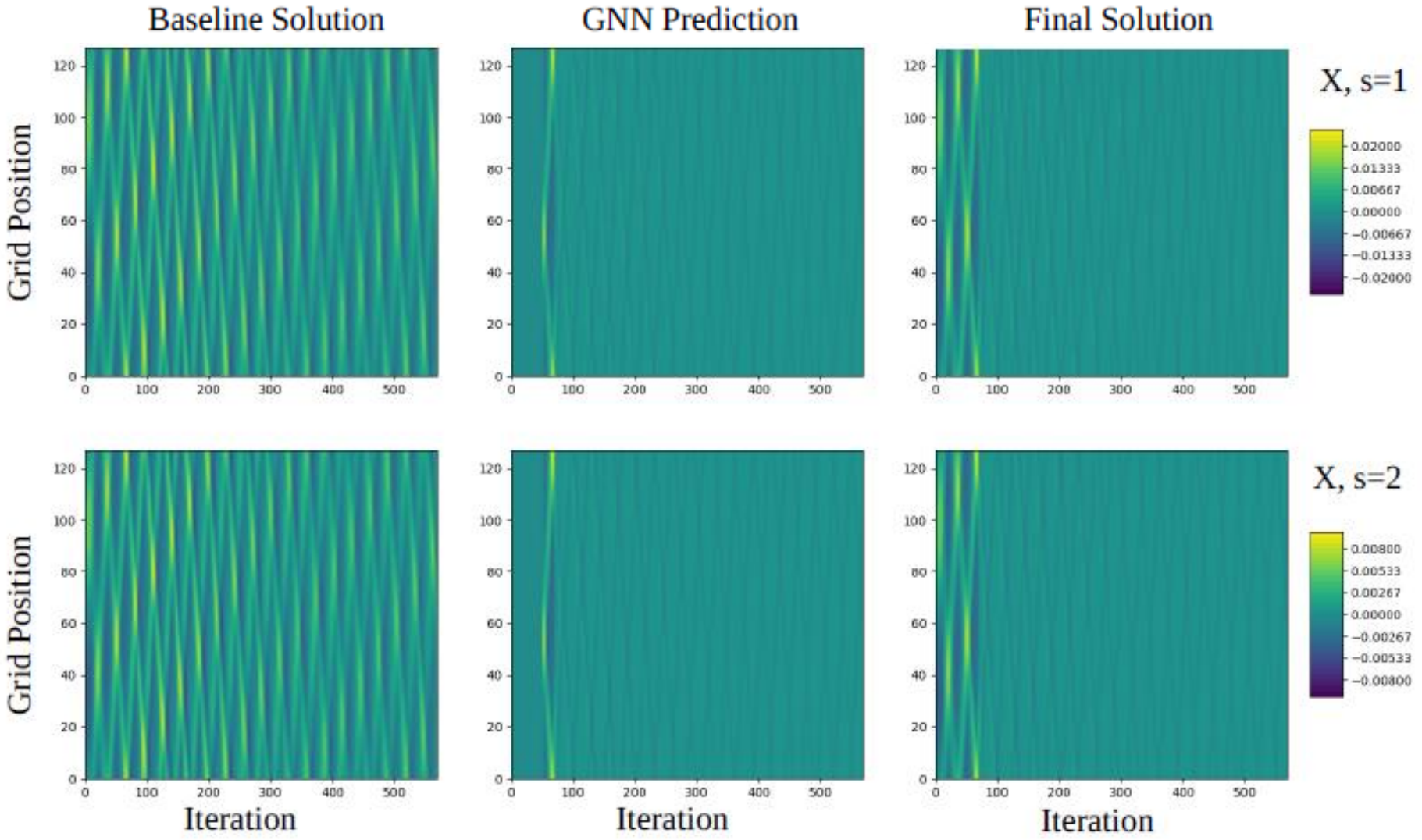
Non-linear warm start reduces the magnitude of the linear data

1D Euler: Contour Plots, Non-Linear + Linear



Results show that the linear GNN functions properly even with the magnitude shift

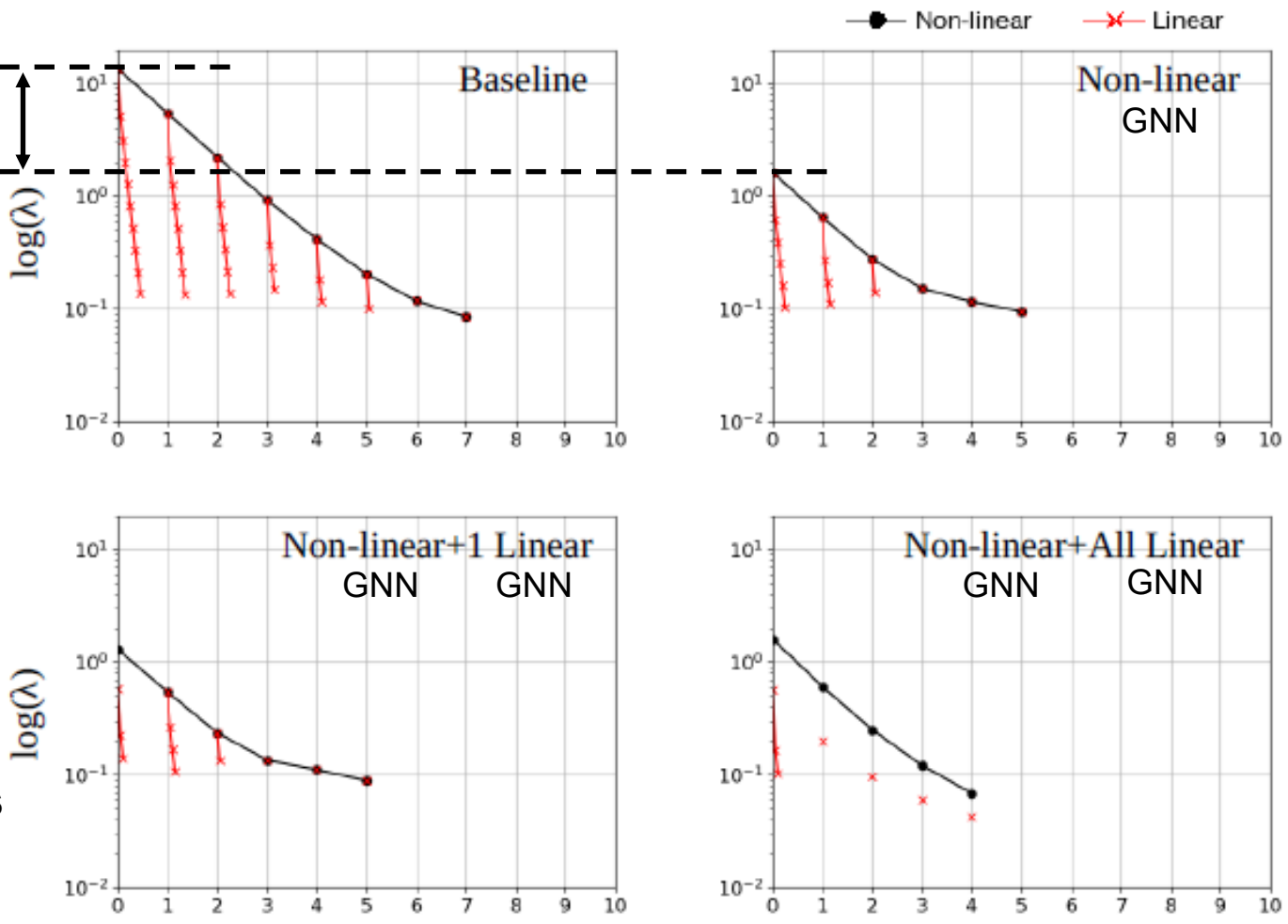
1D Euler: Contour Plots, Non-Linear + Linear



More efficient not to start linear data curation before the non-linear predictions start

1D Euler: Results, Time Step 400

1 Order of Magnitude



8 Nonlinear Iterations
35 Linear Iterations

57.1% savings
6 Nonlinear Iterations
15 Linear Iterations

65.7% savings
6 Nonlinear Iterations
12 Linear Iterations

80.0% savings
5 Nonlinear Iterations
7 Linear Iterations

1D Euler: Conclusions

Nonlinear GNN predictions are reasonable

1 order of magnitude drop in nonlinear residuals

Linear GNN predictions are reasonable for all orders of magnitude

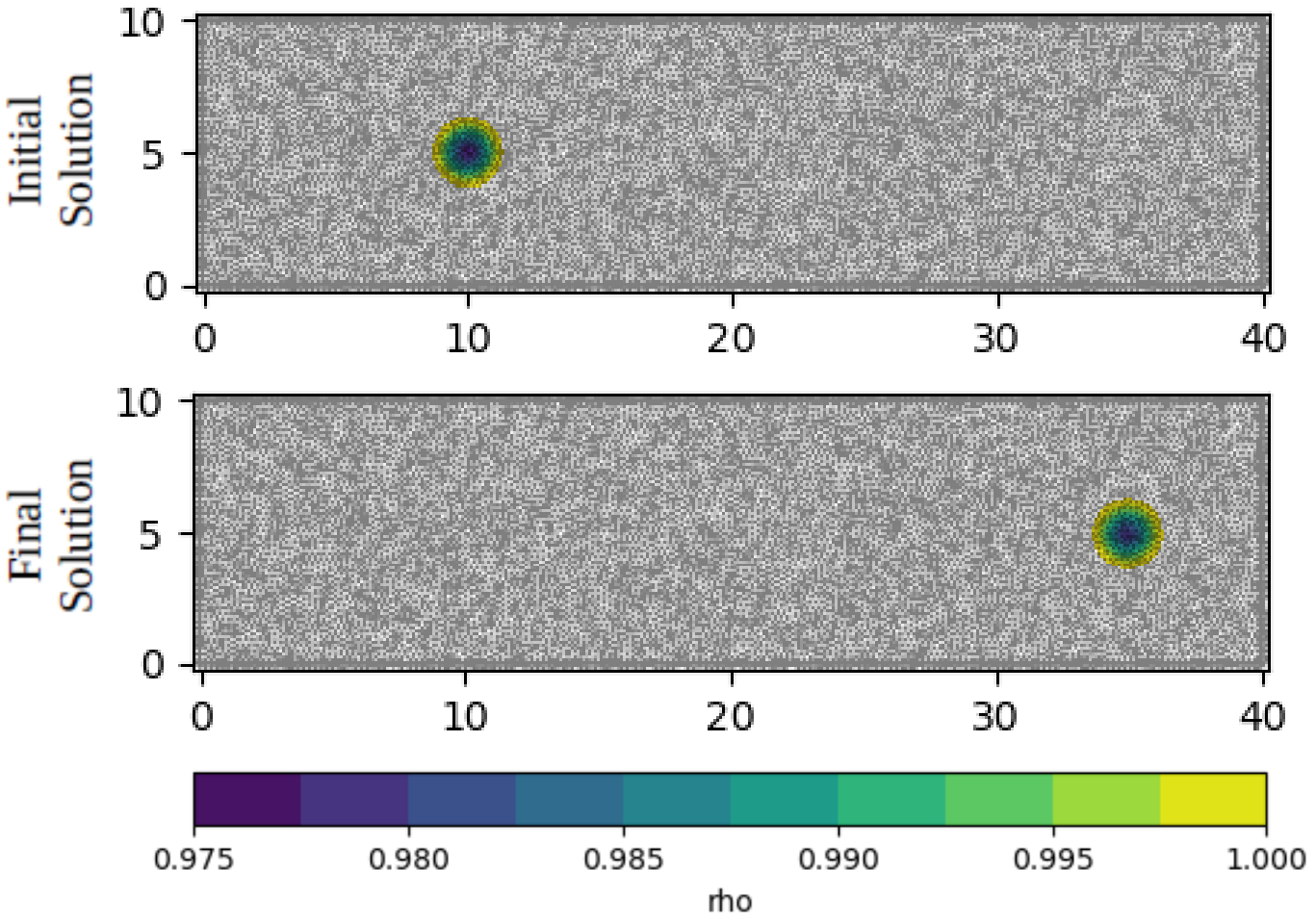
Additional residual drop by adding the linear warm start

This is still a simple problem, but it demonstrates the results we are hoping for even with the additional complexity

2D Euler Vortex: Problem Setup

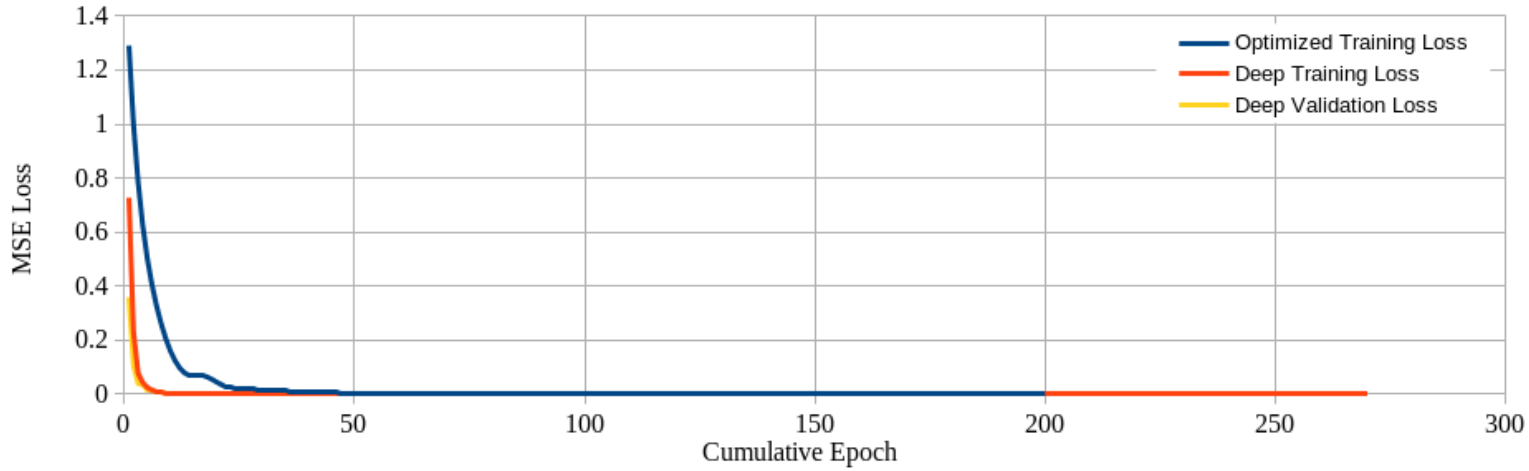
- Vortex propagation
- 2D uniform triangular mesh
- 23K cells

Purpose is to investigate a 2D problem and take a first look at the ML overhead

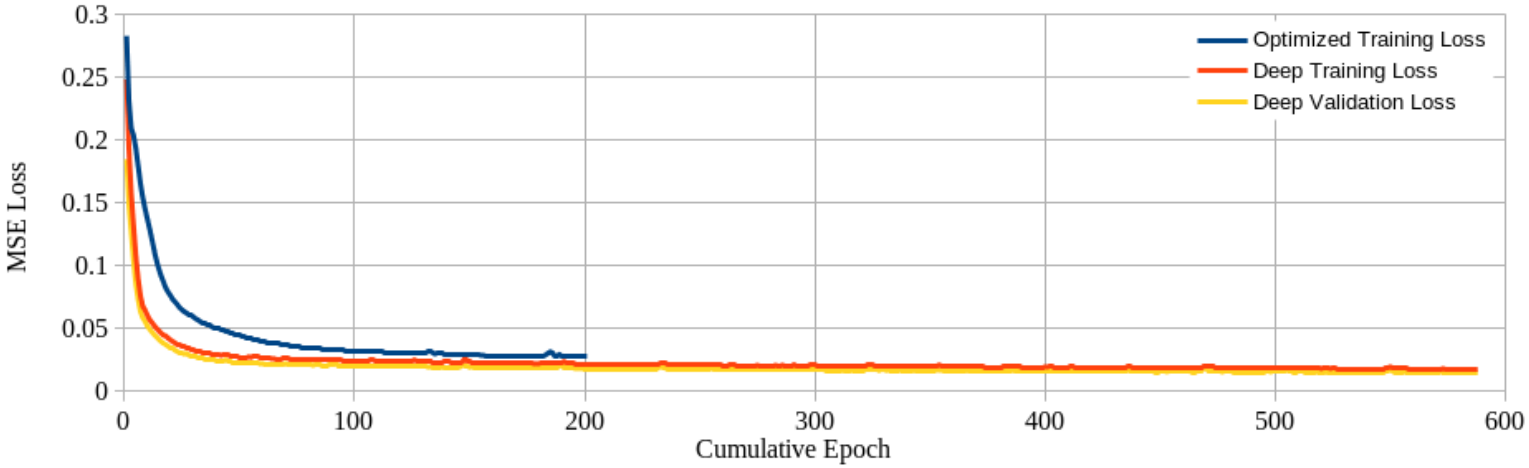


2D Euler Vortex: Model Training Loss

Non-Linear

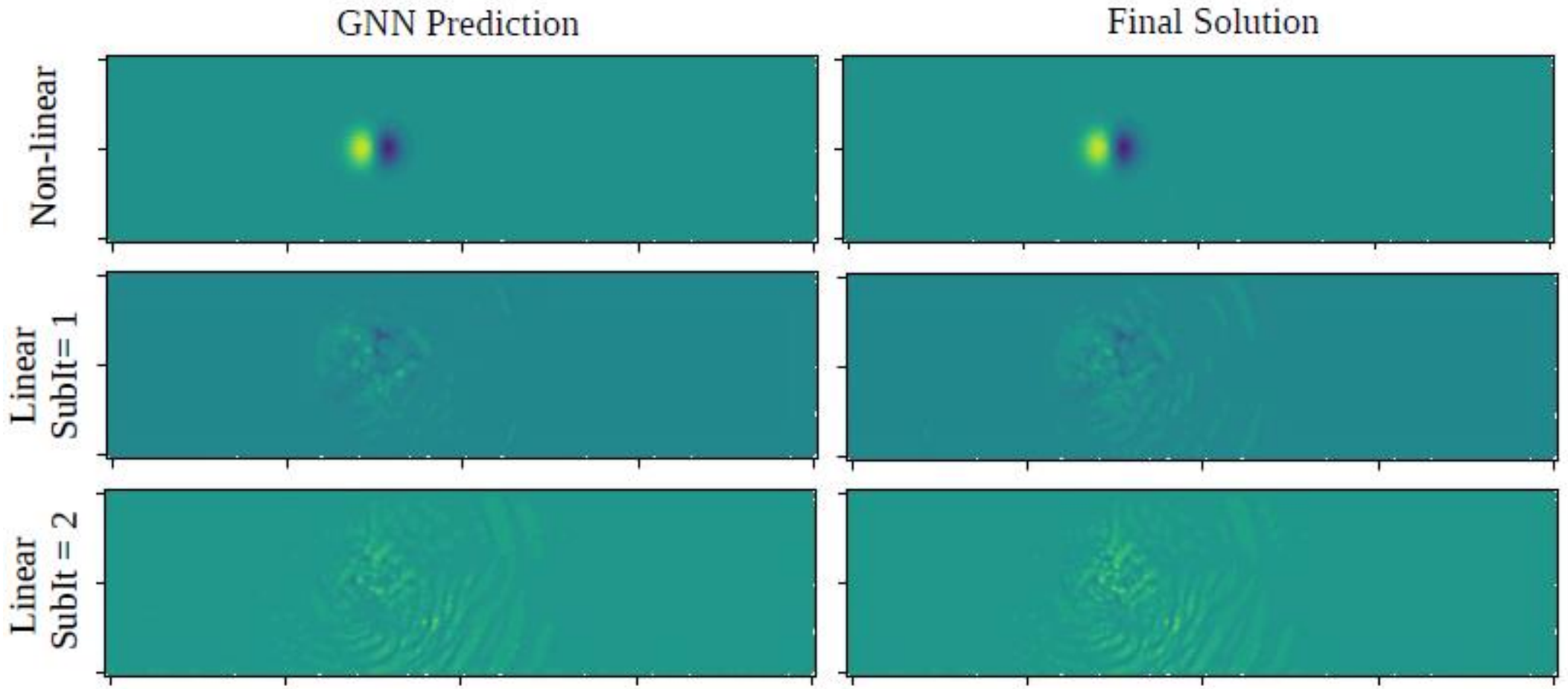


Linear



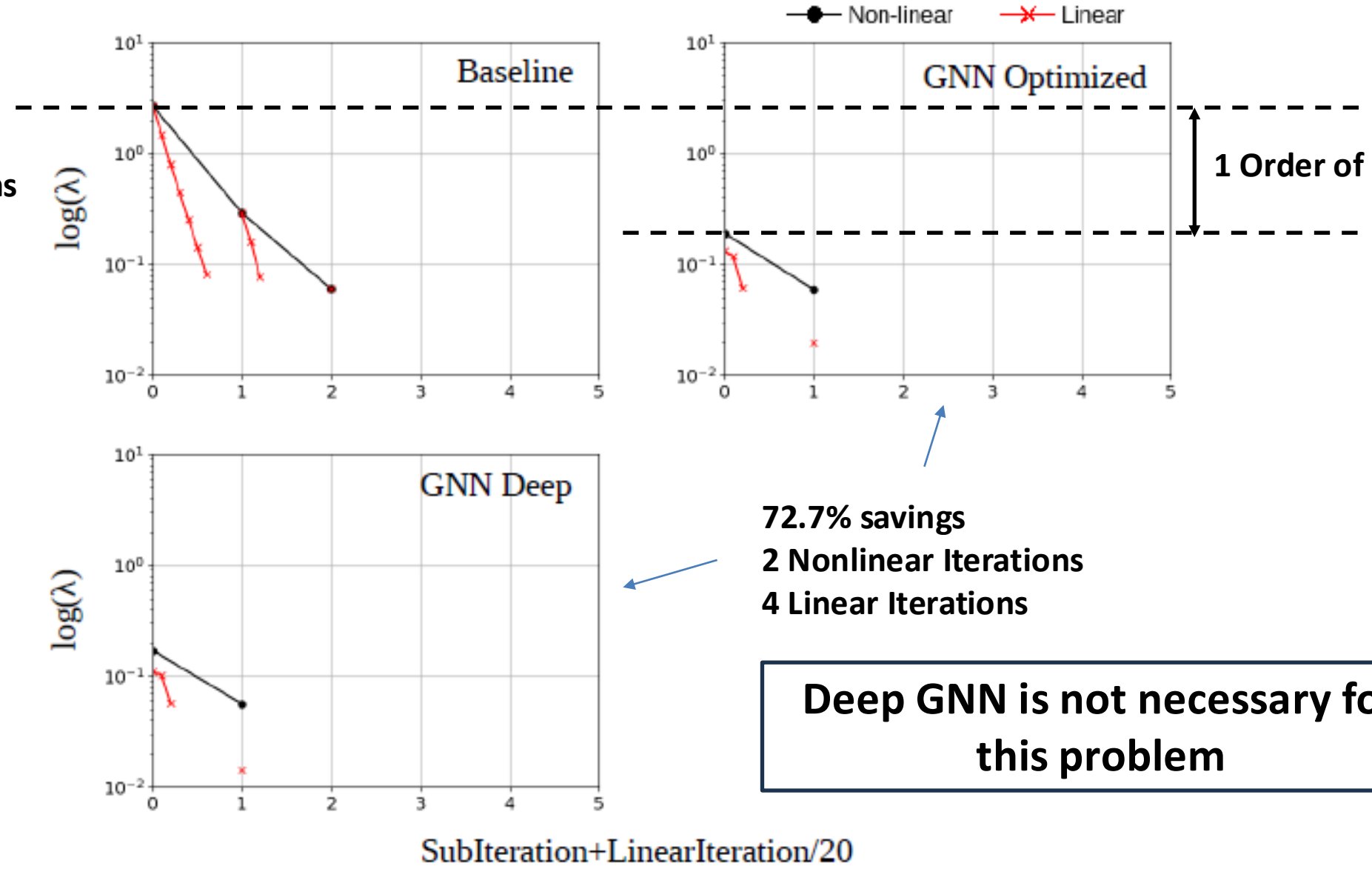
Basically, comparing the performance of more expensive deeply trained models to cheaper "optimized" models

2D Euler Vortex: Contour Plots



2D Euler Vortex: Results, Time Step 400

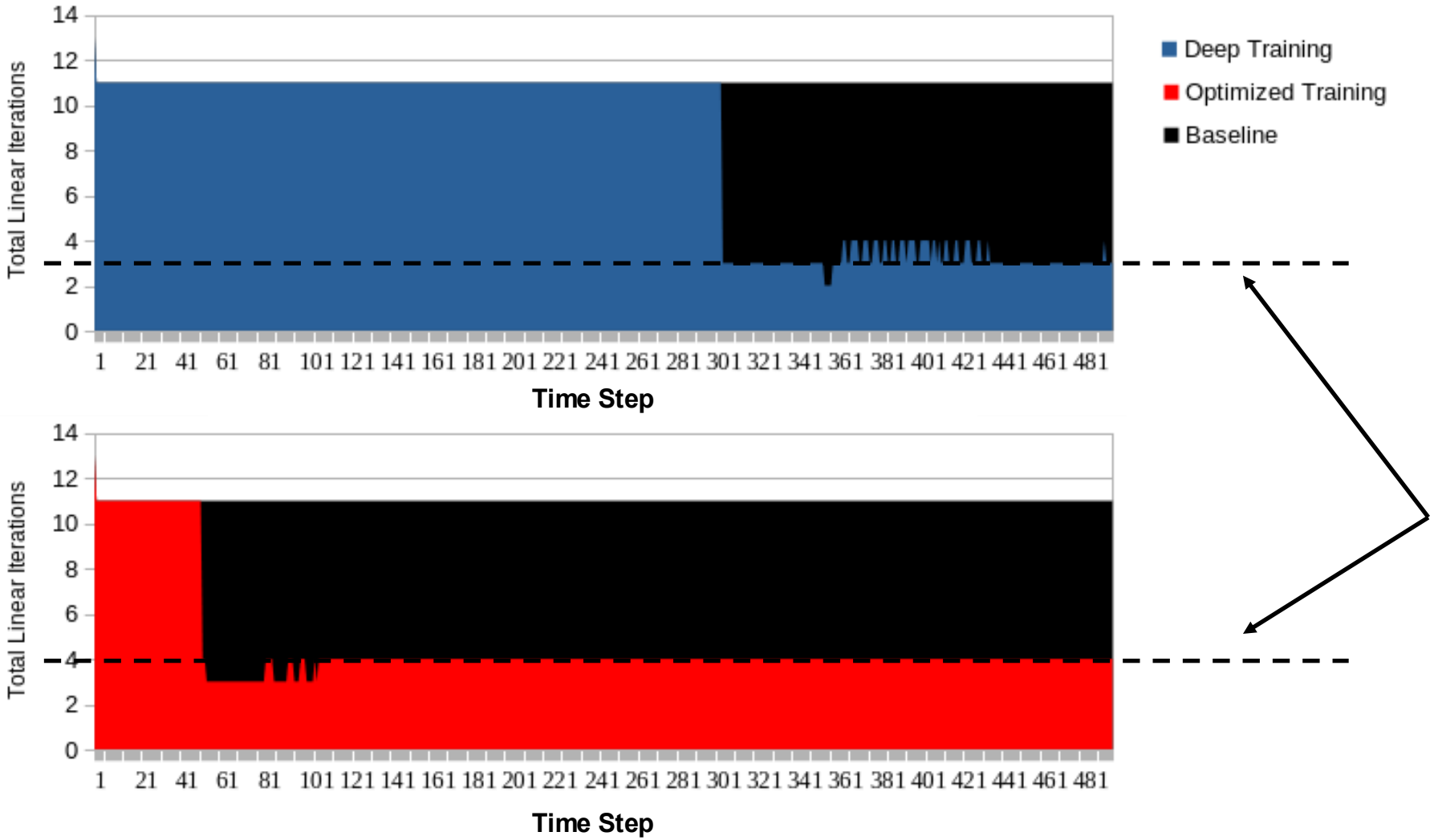
3 Nonlinear Iterations
11 Linear Iterations



72.7% savings
2 Nonlinear Iterations
4 Linear Iterations

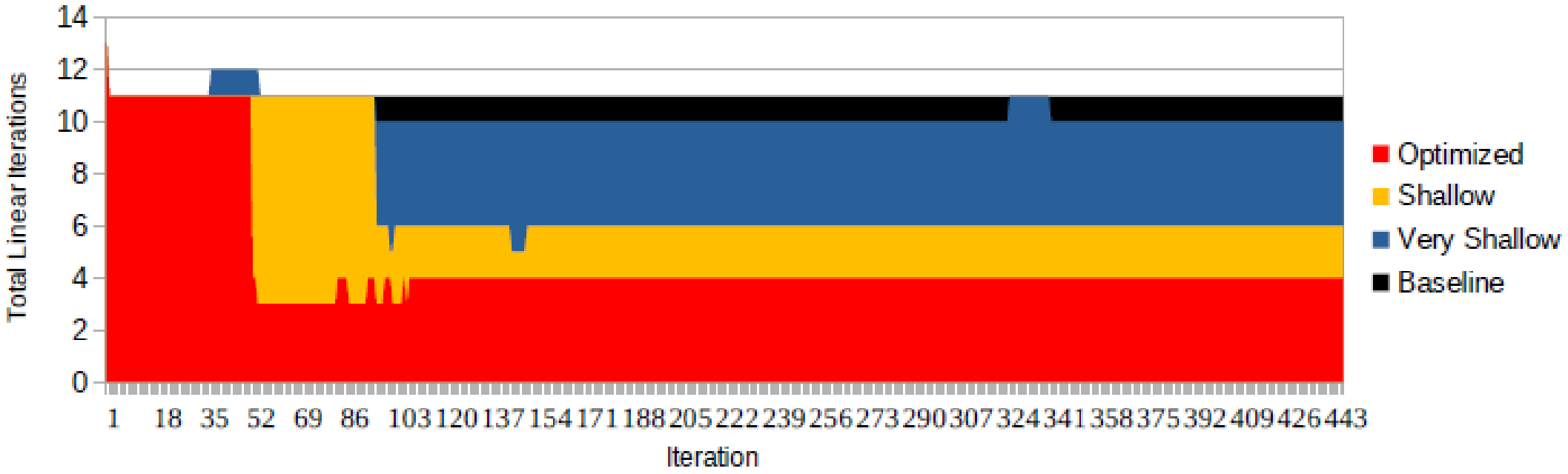
Deep GNN is not necessary for this problem

2D Euler Vortex: Total Linear Iteration Count



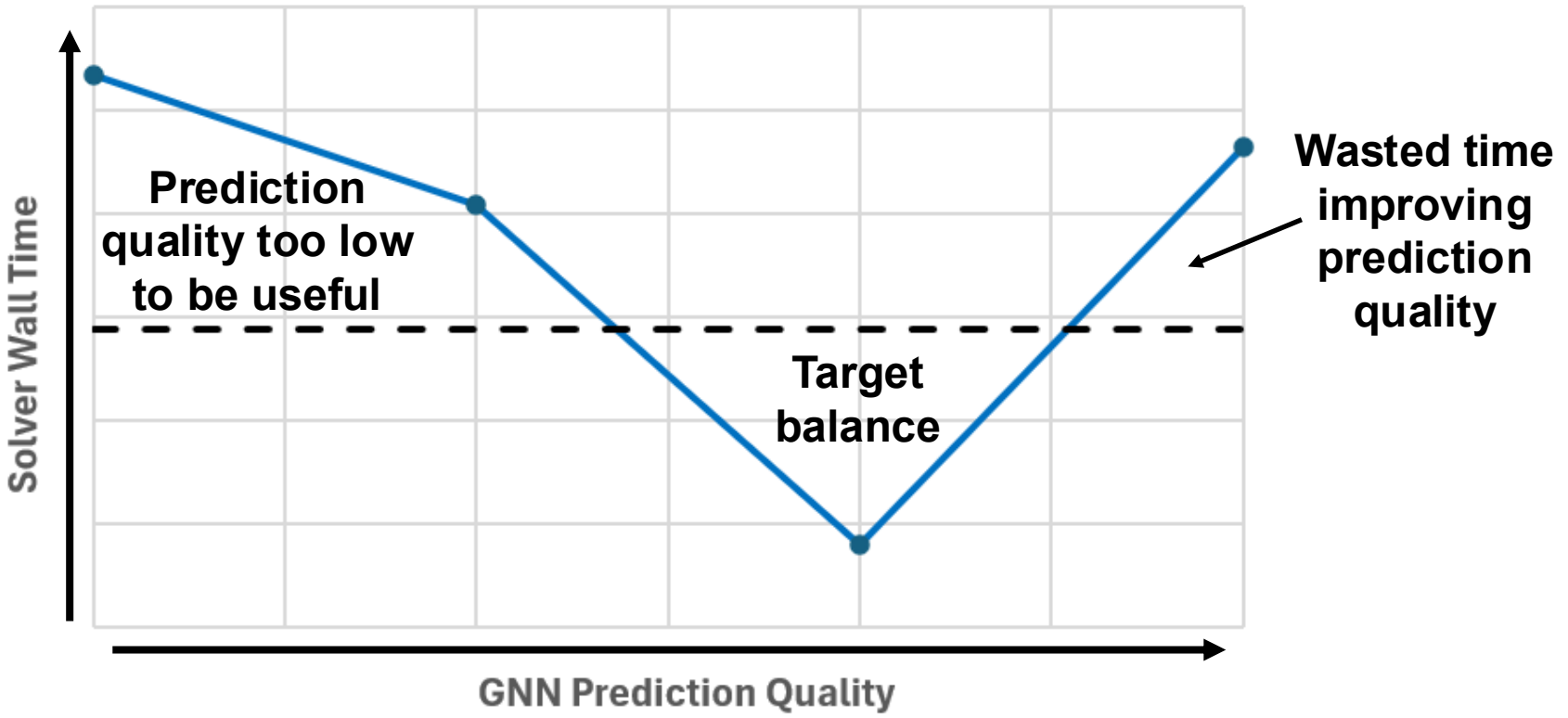
Similar solver savings, but at reduced cost using the optimized "good enough" GNN

2D Euler Vortex: Total Linear Iteration Count, Shallow



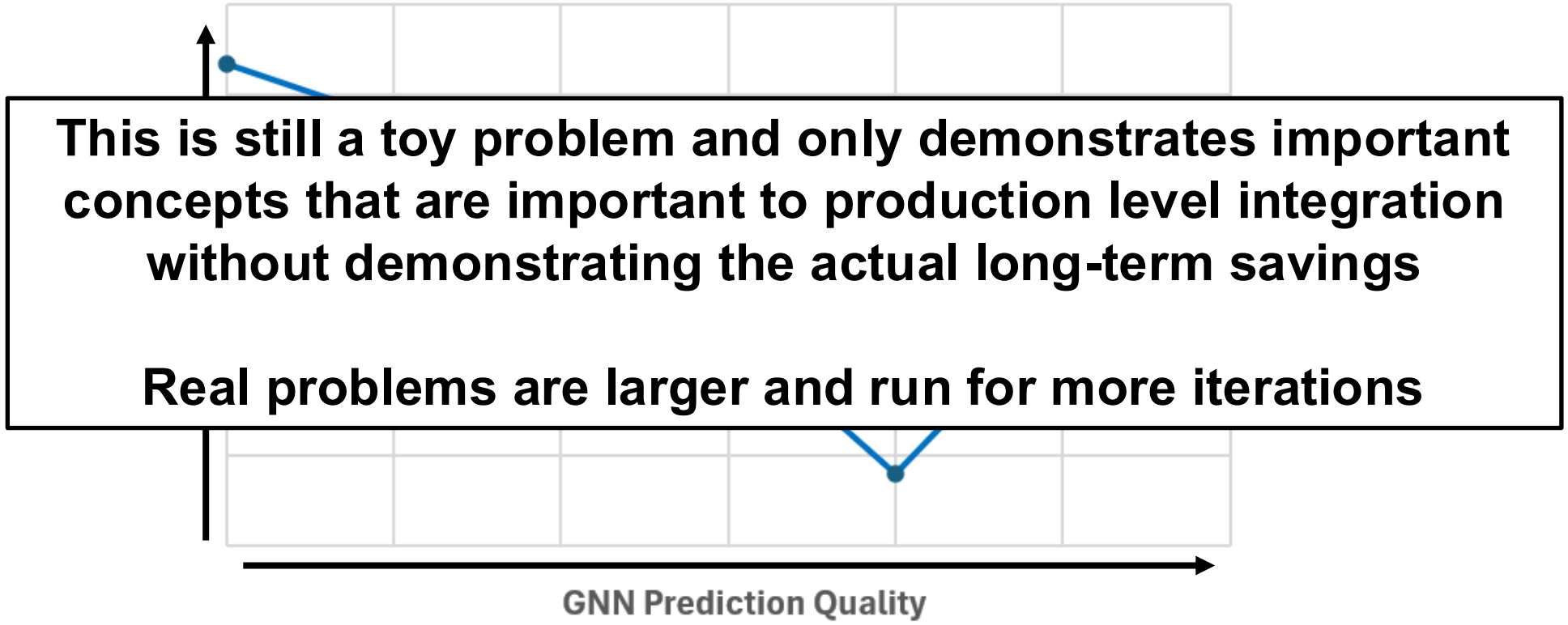
As the GNN prediction accuracy decreases the solver must execute more linear iterations to recover the final solution

2D Euler Vortex: Wall Clock Comparison



Production Level Goal
The method must use a fraction of the time saved in the solver to be effective

2D Euler Vortex: Wall Clock Comparison



This is still a toy problem and only demonstrates important concepts that are important to production level integration without demonstrating the actual long-term savings

Real problems are larger and run for more iterations

Production Level Goal
The method must use a fraction of the time saved in the solver to be effective

2D Euler Vortex: Conclusions

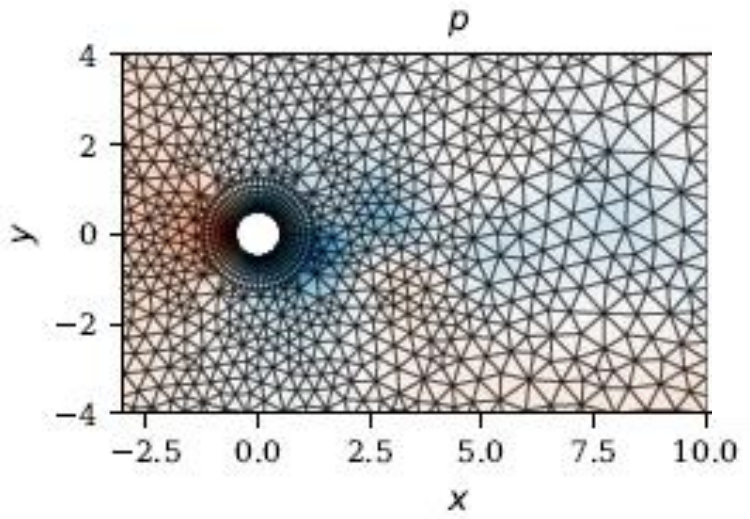
2D results show the same benefits as 1D

Important to balance GNN overhead and solver savings

Demonstrated that the same benefits are obtained in 2D and that when moving to production cases we must carefully examine the balance between solver benefits and time spent training

2D Cylinder: Problem Setup

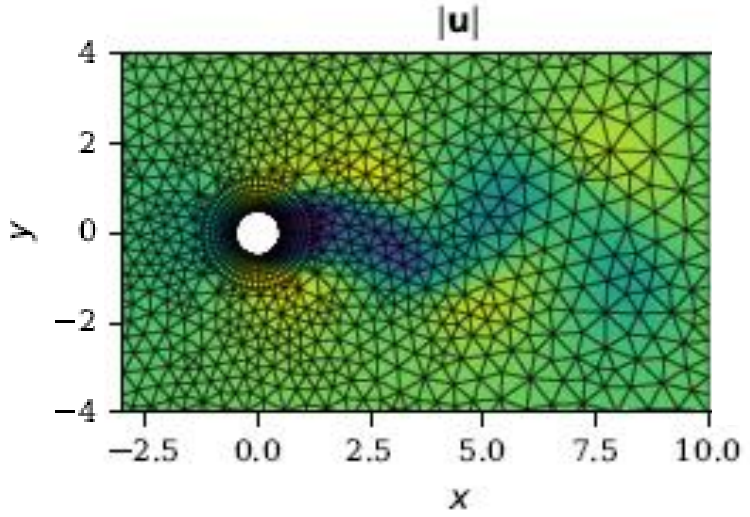
- **Cylinder wall bounded problem**
- **Non-uniform unstructured mesh**
  - Baseline: 5.5K cells
  - Fine: 30K cells
- **Re = 100, unsteady shedding**



Investigating the first wall bounded problem

Compare Jacobi and GMRES methods

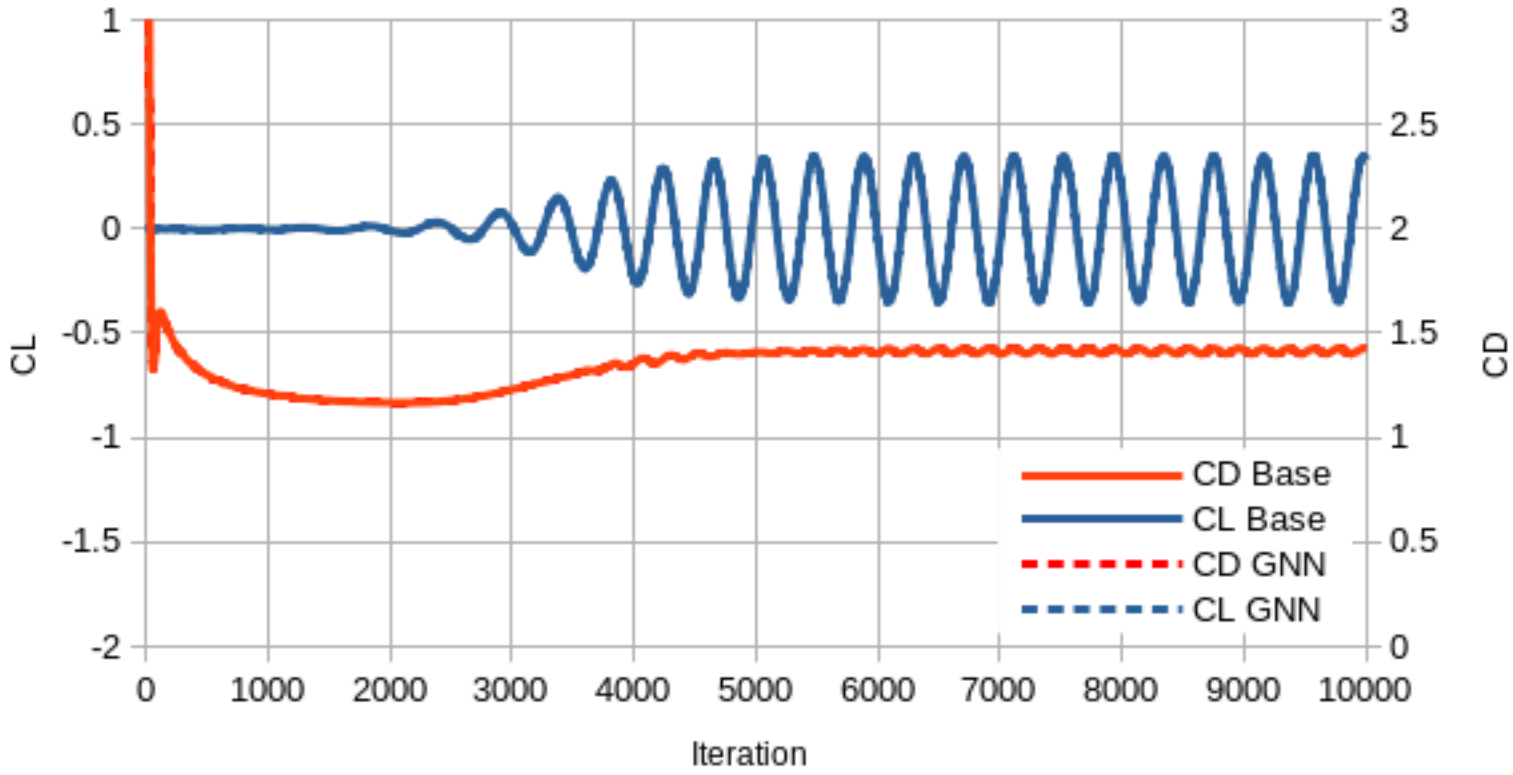
Ported the code from Python to C++
Full GPU operation for the solver and the GNN



Sitaraman J., Abras J., Hosseinverdi, S., and Hariharan N., "Enhancing Production-Oriented Non-Linear PDE Solver Convergence with Learnable Models" ICCFD 13, Milan, Italy, July 6-10, 2026

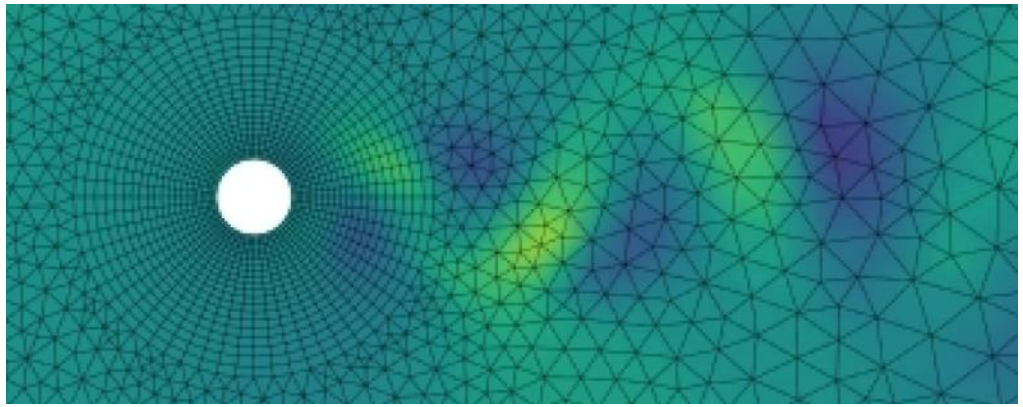
2D Cylinder: Integrated Loads Comparison

- Up until now we have not explicitly demonstrated the impact of the method on the final solution, previous contour plots imply no impact
- This plot explicitly shows that the GNN initialization has little to no impact on the final integrated loads

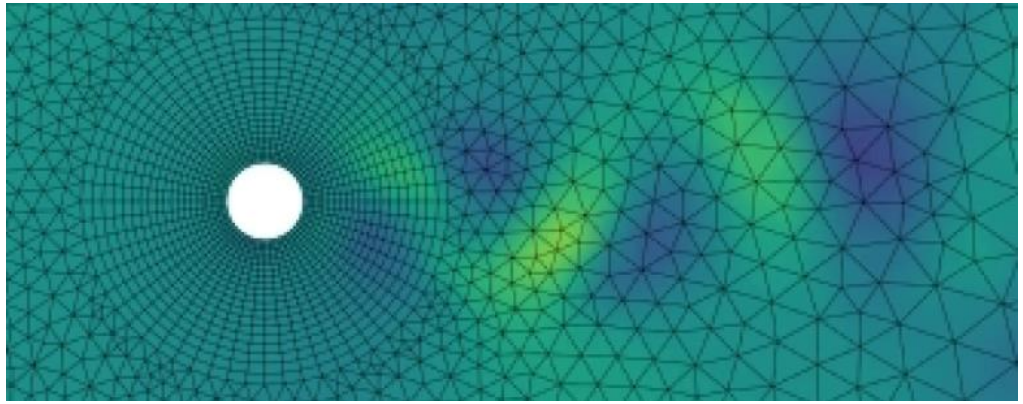


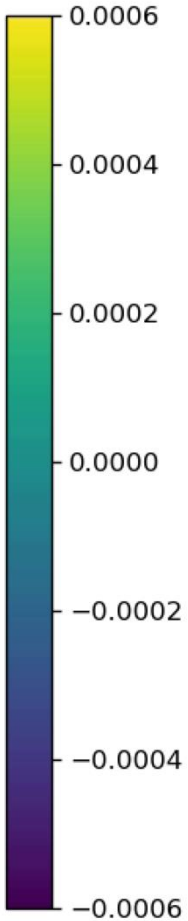
2D Cylinder: Instantaneous Contour Plots

Final/Truth

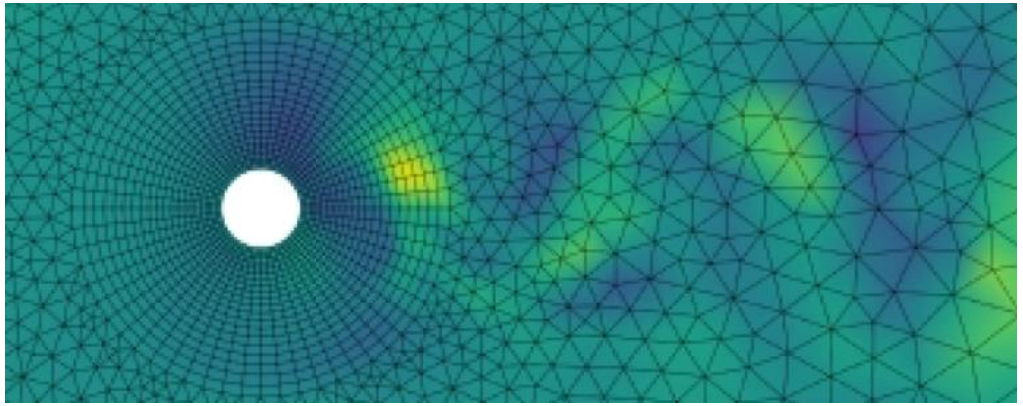


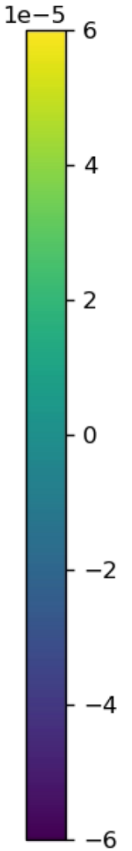
GNN Prediction





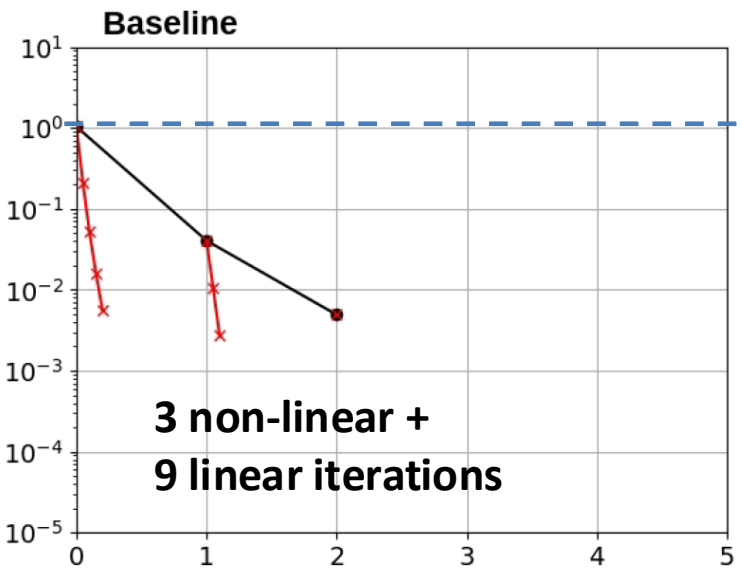
Delta = (GNN Prediction – Truth)

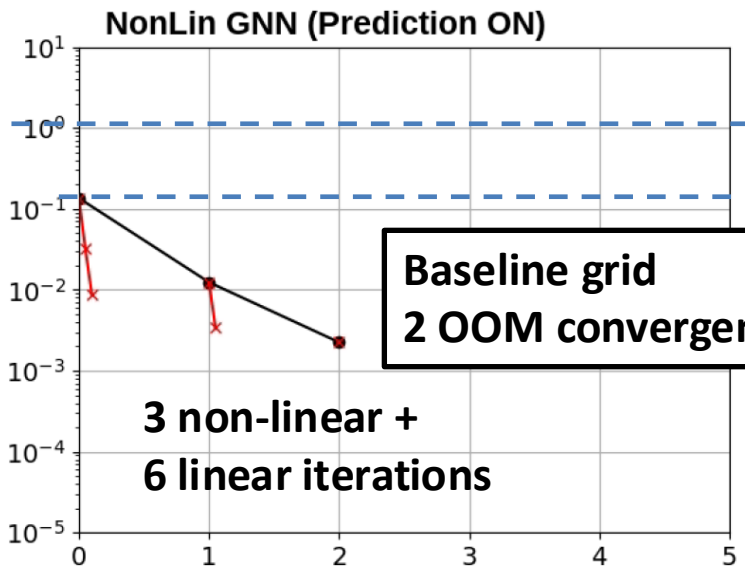




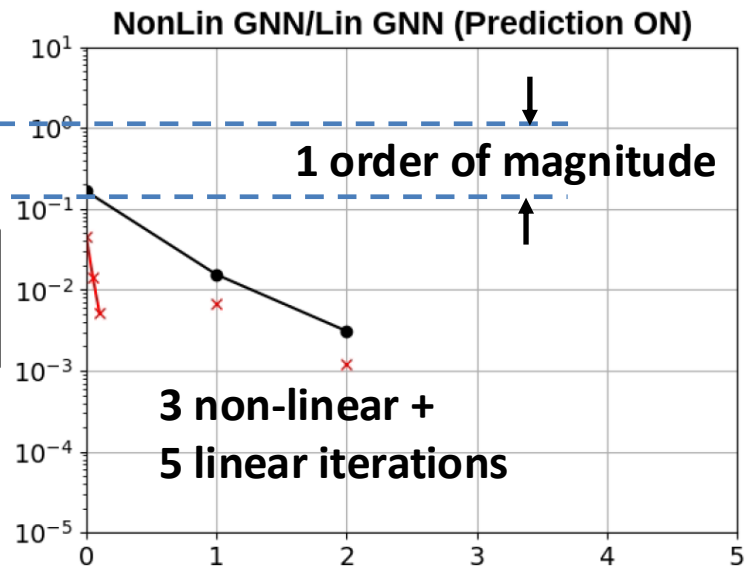
Both baseline and fine grids show similar contour comparisons

2D Cylinder: Jacobi Warm-Start

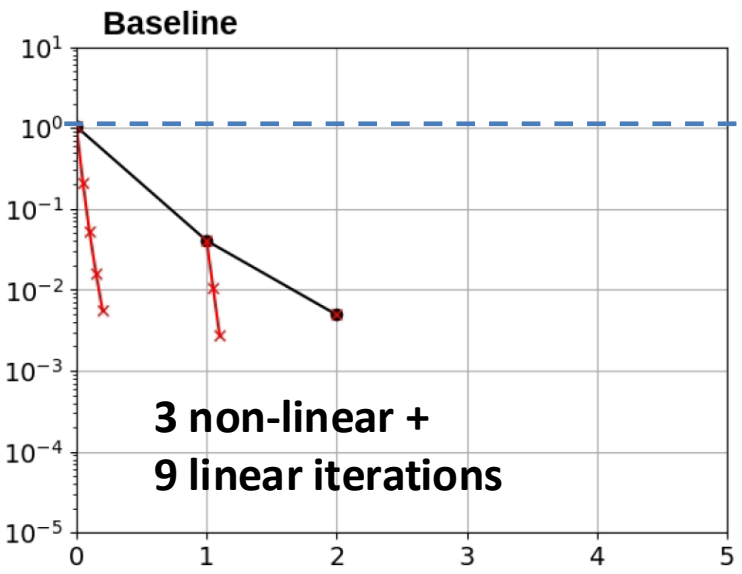


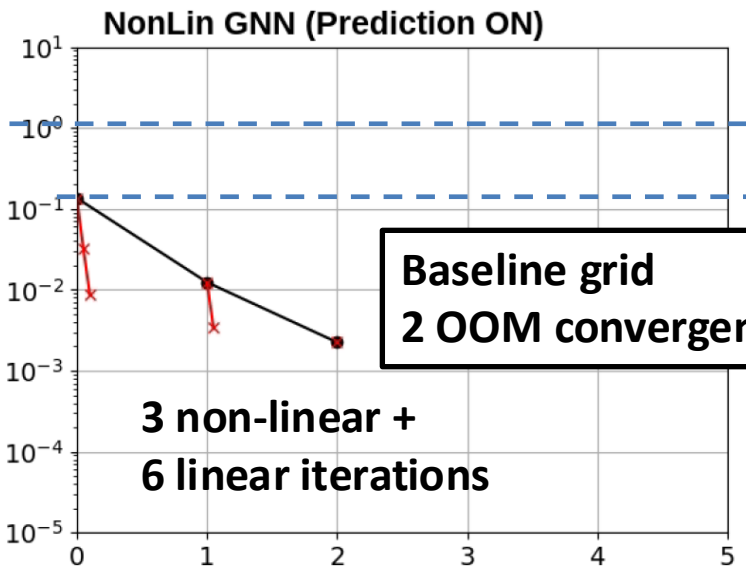


Baseline grid
2 OOM convergence

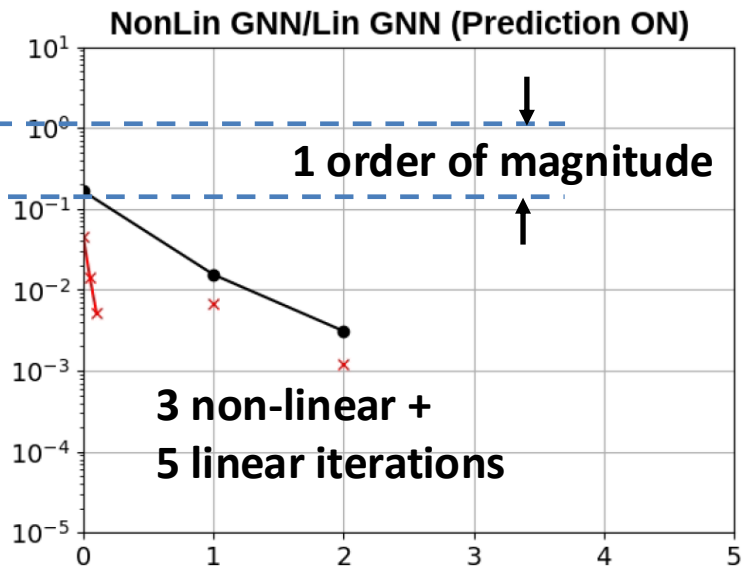


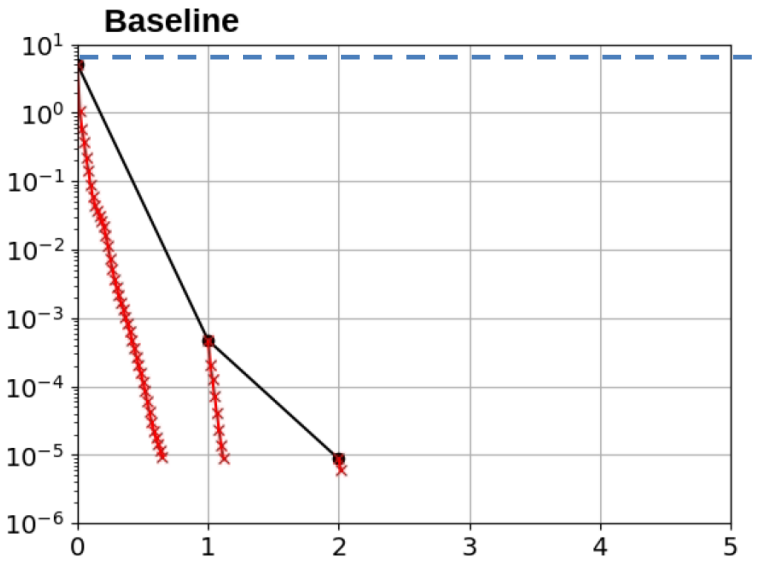
2D Cylinder: Comparing to GMRES Warm-Start

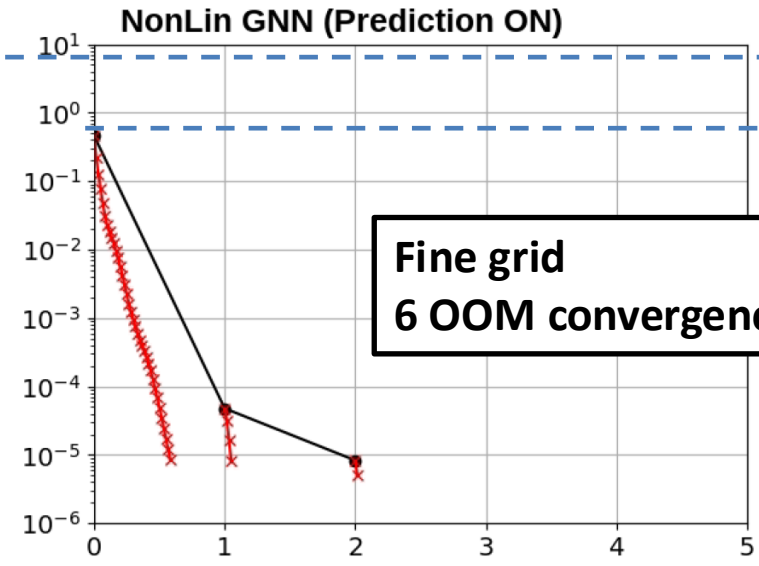




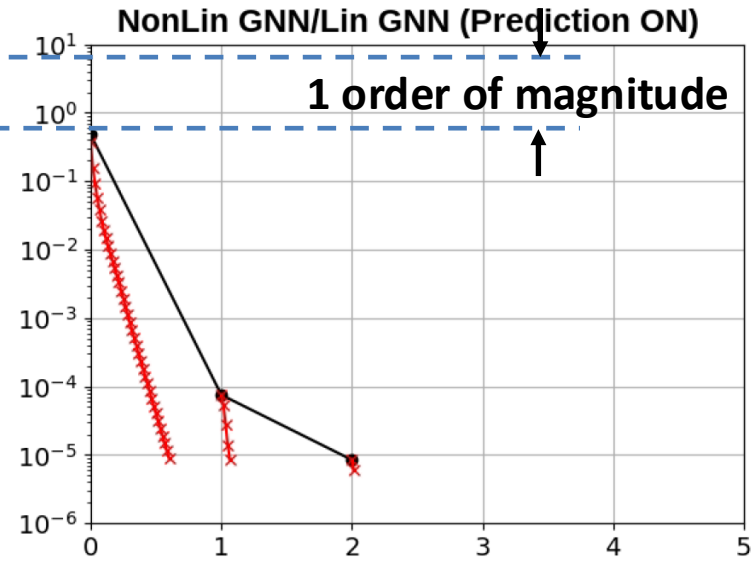
Baseline grid
2 OOM convergence







Fine grid
6 OOM convergence



2D Cylinder: Total Solver Savings

Configuration	Baseline	NonLin GNN	NonLin+Lin GNN	
Coarse grid, Jacobi	209K	150K	126K	39.7% savings
Fine grid, Jacobi	388K	370K	290K	25.3% savings
Fine grid, GMRES	474K	425K	418K	11.8% savings

All cases show some benefit using both the non-linear and linear GNNs

Run-time on a single A100 GPU is only a few minutes

2D Cylinder: Conclusions

Both non-linear and linear models provide improvements

Deep convergence in sub-iterations shows less benefit

Demonstrated that the method works when moving to wall bounded problems

What's Next?

- **Memory:**
  - Memory usage for the 2D cylinder case is greater than the underlying solver
  - Prompted redesign of ML algorithms to test a dataset free method
- **Savings Trend with Complexity:**
  - Look into the addition of edge attributes to the GNN feature set
- **Production Integration:**
  - Integration with the production solver in Helios
  - Compare to the same 2D problems
  - Investigate 3D problems

Abras J., Sitaraman J., Hosseinverdi, S., "Enhancing Production PDE Solver Convergence with Learnable Models," 65th AIAA Aerospace Sciences Meeting, Orlando, FL, January 11-15, 2027

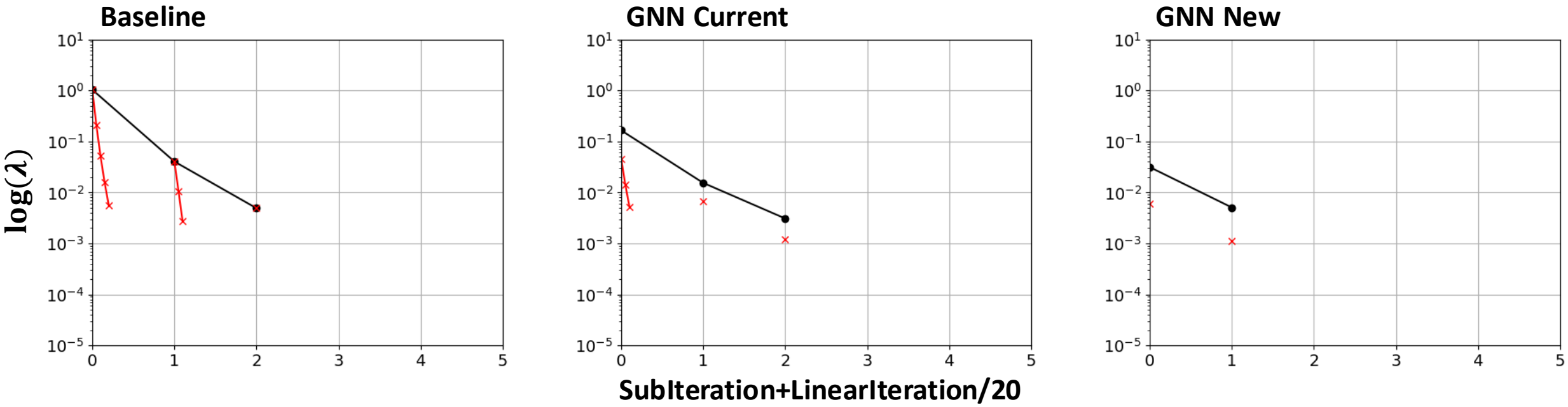
What's Next?

- **Memory:**
  - Memory usage for the 2D cylinder case is greater than the underlying solver
  - Prompted redesign of ML algorithms to test a dataset free method (complete)
- **Savings Trend with Complexity:**
  - Look into the addition of edge attributes to the GNN feature set (complete)
- **Production Integration:**
  - Integration with the production solver in Helios
  - Compare to the same 2D problems
  - Investigate 3D problems

Abras J., Sitaraman J., Hosseinverdi, S., "Enhancing Production PDE Solver Convergence with Learnable Models," 65th AIAA Aerospace Sciences Meeting, Orlando, FL, January 11-15, 2027

Preview: Memory Improvement

Coarse Cylinder, Iteration 25000



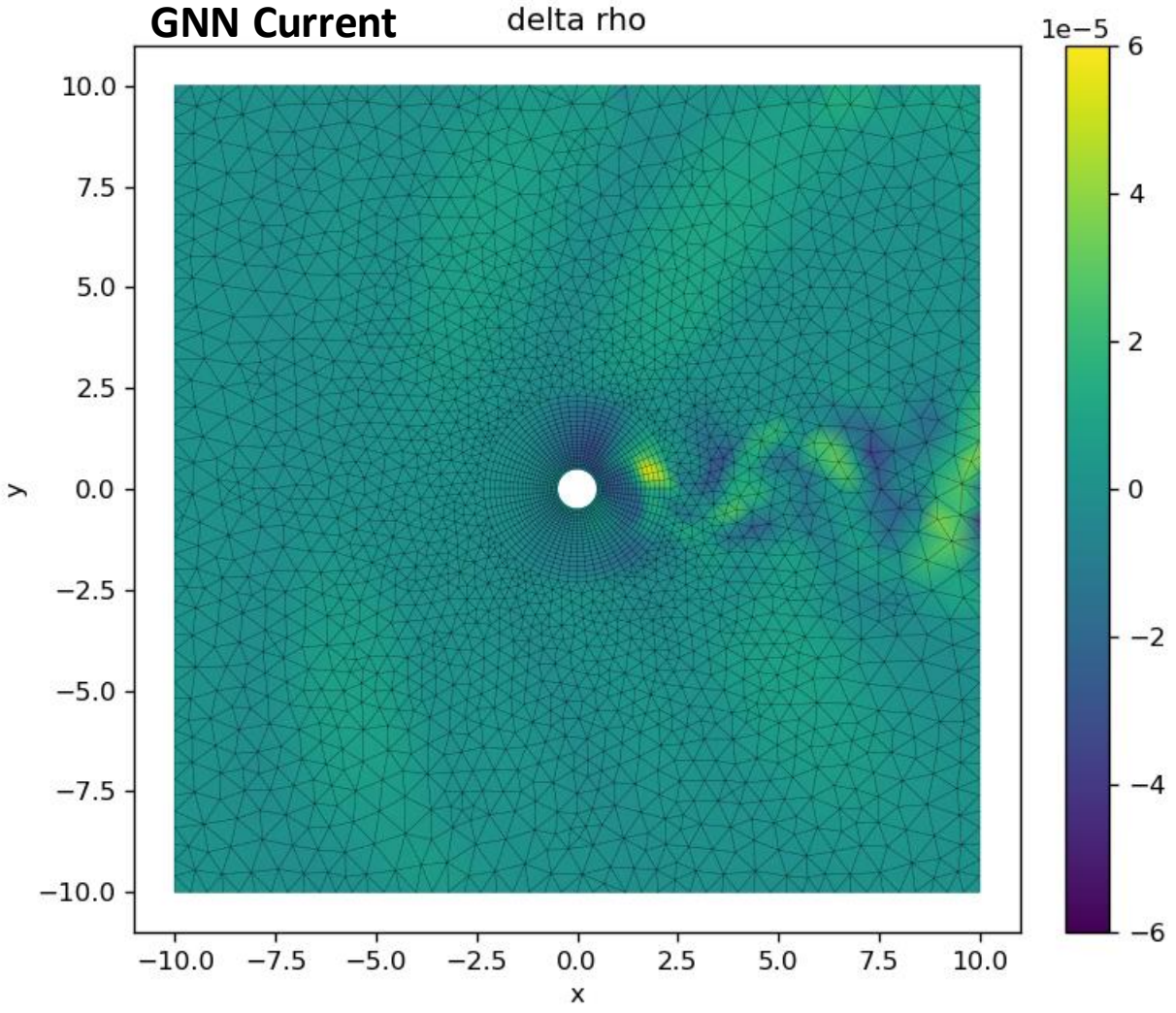
9 Linear Iterations
Memory = 0.512 Gb

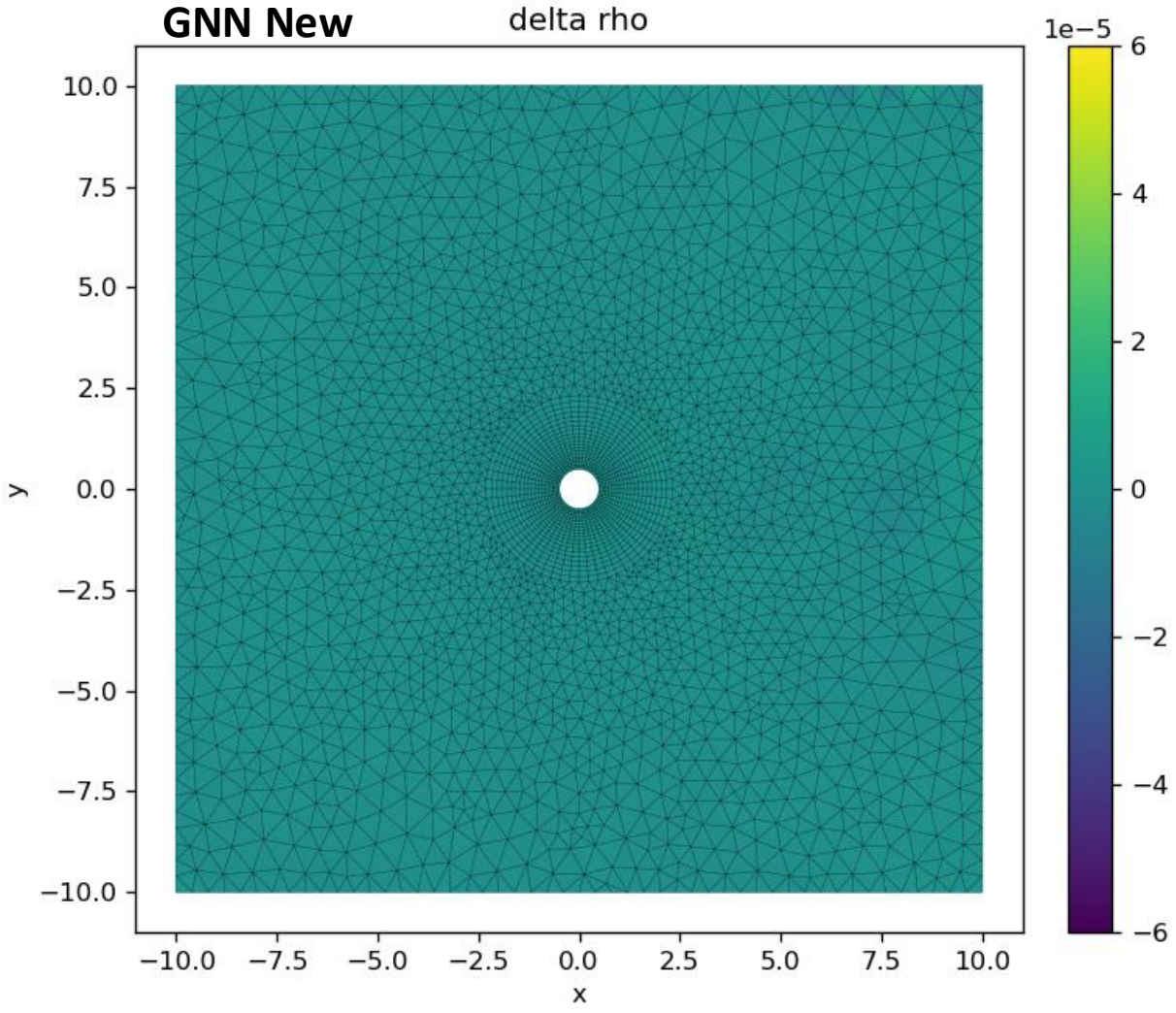
5 Linear Iterations
Savings = 44.4%
Memory = 1.948 Gb

2 Linear Iterations
Savings = 77.8%
Memory = 0.595 Gb

Memory usage has dropped significantly, and the performance of the method has also improved (39.7% total savings to 62.9% total savings)

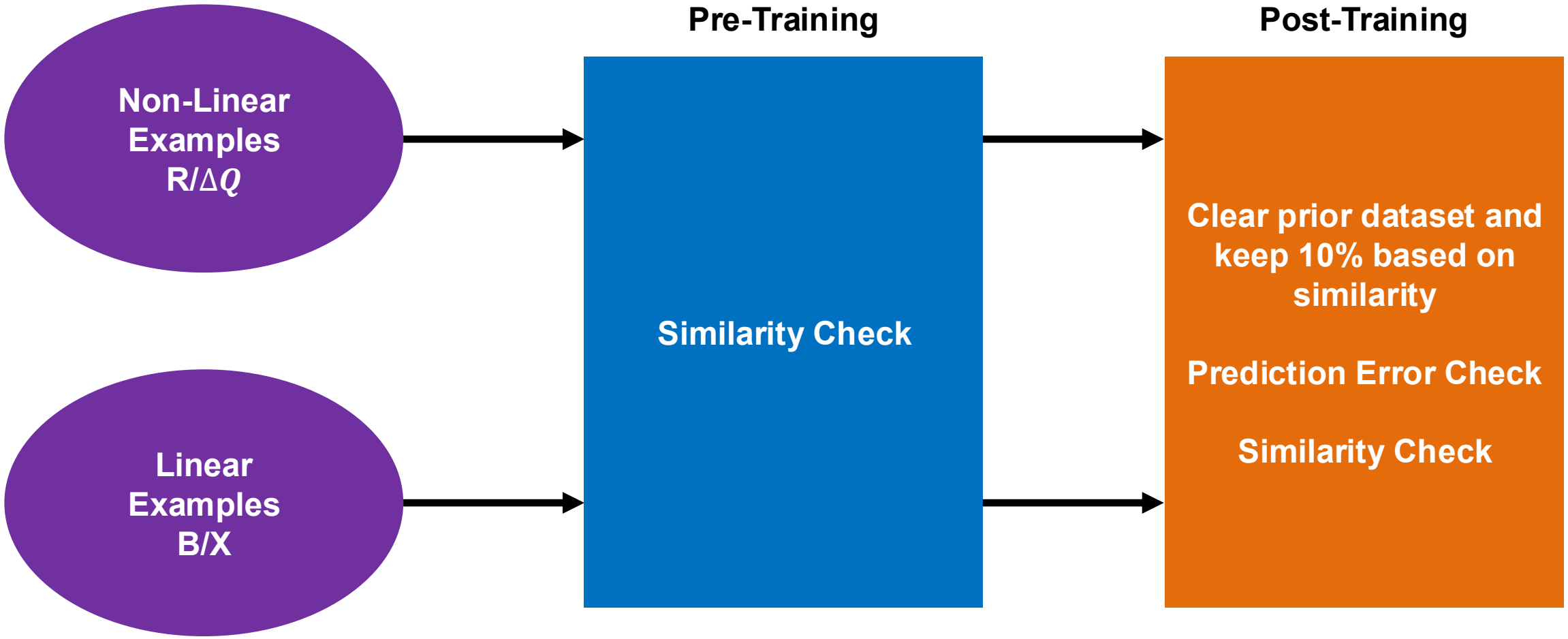
Preview: Prediction Delta





The delta between truth and prediction shows a noticeable improvement (Iteration 25000 shown)

Preview: Core Algorithm Difference, Current



The current version uses a more data-centric method for training

Preview: Core Algorithm Difference, New

Pre-Training

Post-Training

The model is trained on one point at a time

Since no dataset exists the method now uses a skill-based metric with novelty detection to select points for training

Skill = 1.0 perfect prediction
Skill = 0.0 no better than zero initialization
Skill < 0.0 worse than zero initialization

Takes better advantage of the constant presence of truth vs prediction as the simulation runs

More details at SciTech 27

Acknowledgements

Material presented is a product of the CREATE (Computational Research and Engineering for Acquisition Tools and Environments) element of the U.S. Department of Defense HPC Modernization Program. Funding and support for the presented research is also provided by the US Army DEVCOM Aviation and Missile Center.